

PISO-P32C32/P64/C64

User's Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright © 1999 by ICP DAS. All rights are reserved.

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

Tables of Contents

1. INTRODUCTION.....	4
1.1 SPECIFICATIONS.....	4
1.2 ORDER DESCRIPTION.....	5
1.2.1 Options.....	5
1.3 PCI DATA ACQUISITION FAMILY.....	6
1.4 PRODUCT CHECK LIST.....	7
2. HARDWARE CONFIGURATION.....	8
2.1 BOARD LAYOUT.....	8
2.2 ISOLATED D/I ARCHITECTURE.....	11
2.2.1 Sample D/I Architecture for Internal Power.....	13
2.2.2 Sample D/I Architecture for External Power.....	14
2.3 ISOLATED D/O ARCHITECTURE.....	15
2.3.1 Sample D/O Architecture.....	16
2.4 DAUGHTER BOARDS.....	17
2.4.1 DB-37.....	17
2.4.2 DN-37.....	17
2.4.3 DB-8125.....	17
2.5 PIN ASSIGNMENT OF PISO-P32C32.....	18
2.6 PIN ASSIGNMENT OF PISO-P64.....	19
2.7 PIN ASSIGNMENT OF PISO-C64.....	20
3. I/O CONTROL REGISTER.....	21
3.1 HOW TO FIND THE I/O ADDRESS.....	21
3.1.1 PIO_DriverInit.....	23
3.1.2 PIO_GetConfigAddressSpace.....	26
3.1.3 Show_PIO_PISO.....	29
3.2 THE ASSIGNMENT OF I/O ADDRESS.....	30
3.3 ENABLE I/O OPERATION.....	31
3.4 THE I/O ADDRESS MAP.....	31
3.4.1 PISO-P32C32 I/O Mapping.....	32
3.4.2 PISO-P64 I/O Mapping.....	33
3.4.3 PISO-C64 I/O Mapping.....	34
3.4.4 RESET\ Control Register.....	35
3.4.5 AUX Control Register.....	35
3.4.6 AUX data Register.....	36
3.4.7 INT Mask Control Register.....	36

3.4.8	Aux Status Register.....	36
4.	THE APPLICATIONS OF DIGITAL I/O	37
4.1	THE EXAMPLE OF PISO-P32C32	37
4.2	THE EXAMPLE OF PISO-P64	43
4.3	THE EXAMPLE OF PISO-C64	46
5.	DEMO PROGRAM.....	49
5.1	PROGRAM FILE FOR PISO-P32C32	49
5.2	PROGRAM FILE FOR PISO-P64.....	52
5.3	PROGRAM FILE LIST FOR PISO-C64.....	54
5.4	DIAGNOSTIC PROGRAM.....	56
5.4.1	Diagnostic program for DOS	56
5.4.2	Diagnostic program for WINDOWS.....	56
5.5	DEMO PROGRAM FOR PISO-P32C32	57
5.5.1	DEMO1 for PISO-P32C32.....	58
5.5.2	DEMO2 for PISO-P32C32.....	59
5.5.3	DEMO3 for PISO-P32C32.....	61
5.6	DEMO PROGRAM FOR PISO-P64	63
5.6.1	DEMO1 for PISO-P64	63
5.7	DEMO PROGRAM FOR PISO-C64.....	65
5.7.1	DEMO1 for PISO-C64	65

1. Introduction

The PISO-P32C32 consists of 32 channels of isolated D/I & 32 channels of isolated D/O. The PISO-P64 consists of 64 channels of isolated D/I. The PISO-C64 consists of 64 channels of isolated D/O. The D/I specifications of PISO-P32C32 & PISO-P64 are the same. The D/O specifications of PISO-P32C32 & PISO-C64 are the same.

1.1 Specifications

Isolated digital input

- Input voltage: 3.5V to 30V
- Input impedance: 1.2K
- Isolation voltage:
Using internal power: 3000V
Using external power: 3750V
- Response time: 10K Hz max.

Isolated digital output

- Isolation voltage: 3750V
- Open collector output: 125 mA/30V per channel
- Response time: 1K Hz typical

I/O channels

	D/I channels	D/O channels
PISO-P32C32	32	32
PISO-P64	64	0
PISO-C64	0	64

Other specifications

- PC compatible PCI bus
- Four isolated I/O bank
- Operating Temperature: 0°C to 60°C
- Storage Temperature: -20°C to 80°C
- Humidity: 0 to 90% non-condensing
- Dimensions:
 - PISO-P32C32: 180mm X 105mm
 - PISO-P64: 180mm X 105mm
 - PISO-C64: 180mm X 105mm
- Power Consumption:
 - PISO-P32C32: +5V @ 600mA (typical)
 - PISO-P64: +5V @ 400mA (typical)
 - PISO-C64: +5V @ 800mA (typical)

1.2 Order Description

- PISO-P32C32: PCI bus D/I of 32-bit, D/O of 32-bit.
- PISO-P64: PCI bus D/I of 64-bit.
- PISO-C64: PCI bus D/O of 64-bit.

1.2.1 Options

- DB-24P, DB-24PD: 24 channel isolated D/I board
- DB-24R, DB-24RD: 24 channel relay board
- DB-24PR, DB-24PRD: 24 channel power relay board
- DB-16P8R: 16 channel isolated D/I and 8 channel relay output board
- DB-24POR: 24 channel Photo MOS output board
- DB-24SSR: 24 channel Solid State output board
- DB-24C: 24-channel open-collector output board
- ADP-37/PCI: extender, 50-pin OPTO-22 header to DB-37 for PCI Bus I/O boards
- ADP-50/PCI: extender, 50-pin OPTO-22 header to 50-pin header, for PCI Bus I/O boards
- NAPPCI/win: DLLs for Windows 95/98
- NAPPCI/wnt: DLLs for Windows NT 4.0
- NAPVIEW/1: LabVIEW driver for Windows 95/98
- NAPVIEW/2: LabVIEW driver for Windows NT

1.3 PCI Data Acquisition Family

We provide a family of PCI-BUS data acquisition cards. These cards can be divided into three groups as following:

1. PCI-series: first generation, isolated or non-isolated cards

PCI-1002/1202/1800/1802/1602: multi-function family, non-isolated

PCI-P16R16/P16C16/P16POR16/P8R8: D/I/O family, isolated

PCI-TMC12: timer/counter card, non-isolated

2. PIO-series: cost-effective generation, non-isolated cards

PIO-823/821: multi-function family

PIO-D144/D96/D64/D56/D48/D24: D/I/O family

PIO-DA16/DA8/DA4: D/A family

3. PISO-series: cost-effective generation, isolated cards

PISO-813: A/D card

PISO-P32C32/P64/C64: D/I/O family

PISO-P8R8/P8SSR8AC/P8SSR8DC: D/I/O family

PISO-730: D/I/O card

PISO-DA2: D/A card

1.4 Product Check List

In addition to this manual, the package includes the following items:

- PISO-P32C32/P64/C64 card.
- One piece of company floppy diskette or CD-ROM.
- One piece of release note.

It's recommended to read the release note first. All important information will be given in release notes as follows:

1. Where you can find the software driver & utility.
2. How to install software & utility.
3. Where is the diagnostic program.
4. FAQ.

Attention !

If any of these items is missing or damaged, contact the dealer from whom you purchased the product. Save the shipping materials and carton in case you want to ship or store the product in the future.

2. Hardware configuration

2.1 Board Layout

The board layout of PISO-P32C32 is given as follows:

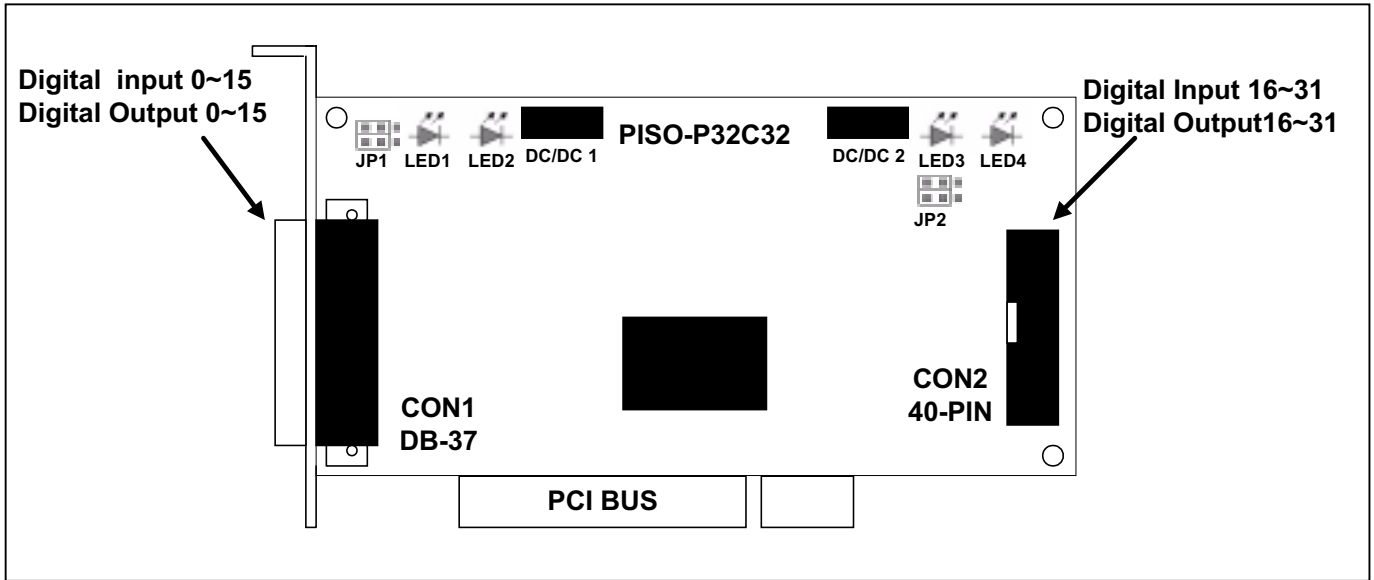




Figure 2-1A. Board layout of PISO-P32C32

JP1/JP2	 INTERNAL	 EXTERNAL
	Default settling	

LED1: Power indicator for DO_0 to DO_15

LED2: Power indicator for DI_0 to DI_15

LED3: Power indicator for DO_16 to DO_31

LED4: Power indicator for DI_16 to DI_31

JP1: Select internal/external power for DI_0 to DI_15 (3000V isolation)

JP2: Select internal/external power for DI_16 to DI_31 (3000V isolation)

Isolation bank 1: DI_0 to DI_15, Power=CON1_18,Ground=CON1_19

Isolation bank 2: DO_0 to DO_15, Power=CON1_37,Ground=CON1_1 & CON1_20

Isolation bank 3: DI_16 to DI_31, Power=CON2_18,Ground=CON2_19

Isolation bank 4: DO_16 to DO_31, Power=CON2_37,Ground=CON2_1 & CON2_20

All these four banks are fully isolated from each other.

The board layout of PISO-P64 is given as follows:

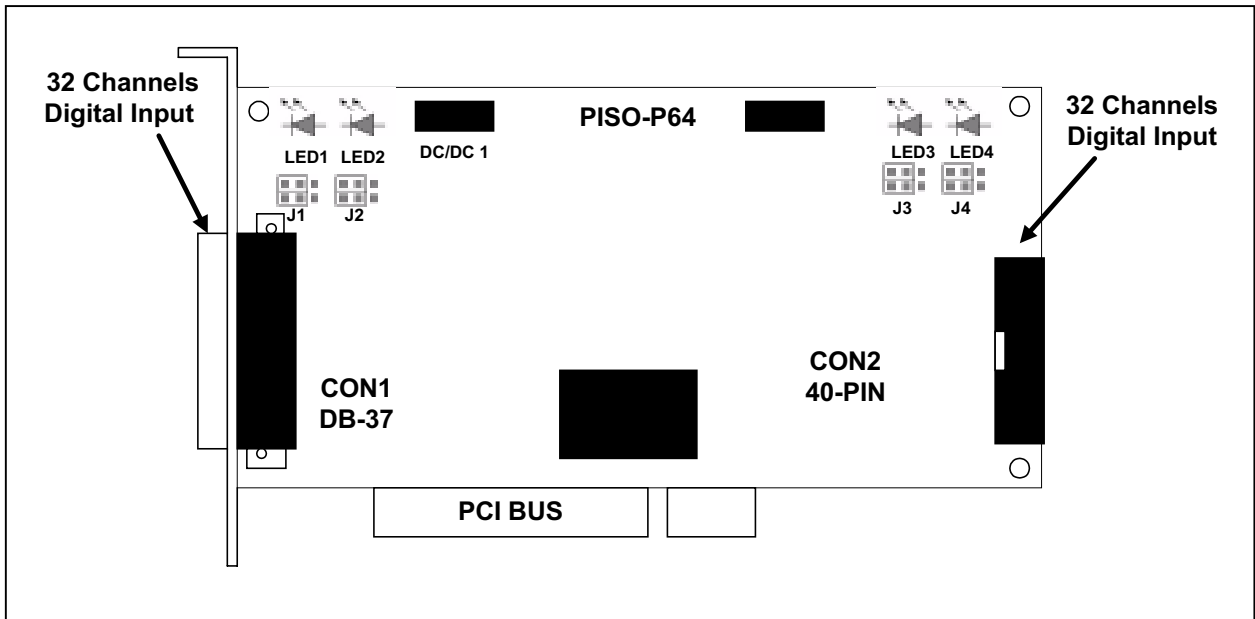




Figure 2-1B. Board layout of PISO-P64

J1/J2/J3/J4	 INTERNAL	 EXTERNAL
	Default setting	

- LED 1: power indicator for DI_0 to DI_15
- LED 2: power indicator for DI_16 to DI_31
- LED 3: power indicator for DI_32 to DI_47
- LED 4: power indicator for DI_48 to DI_63

- J1: select internal/external power for DI_0 to DI_15 (3000V isolation)
- J2: select internal/external power for DI_16 to DI_31 (3000V isolation)
- J3: select internal/external power for DI_32 to DI_47 (3000V isolation)
- J4: select internal/external power for DI_48 to DI_63 (3000V isolation)

Isolation bank 1: DI_0 to DI_15, Power=CON1_18, Ground=CON1_1
 Isolation bank 2: DI_16 to DI_31, Power=CON1_37, Ground=CON1_20
 Isolation bank 3: DI_32 to DI_47, Power=CON2_18, Ground=CON2_1
 Isolation bank 4: DI_48 to DI_63, Power=CON2_37, Ground=CON2_20
 All these four banks are fully isolated from each other.

The DC/DC1 is used to provide the internal power supply for bank1 & bank2.
 The DC/DC2 is used to provide the internal power supply for bank3 & bank4.

The board layout of PISO-C64 is given as follows:

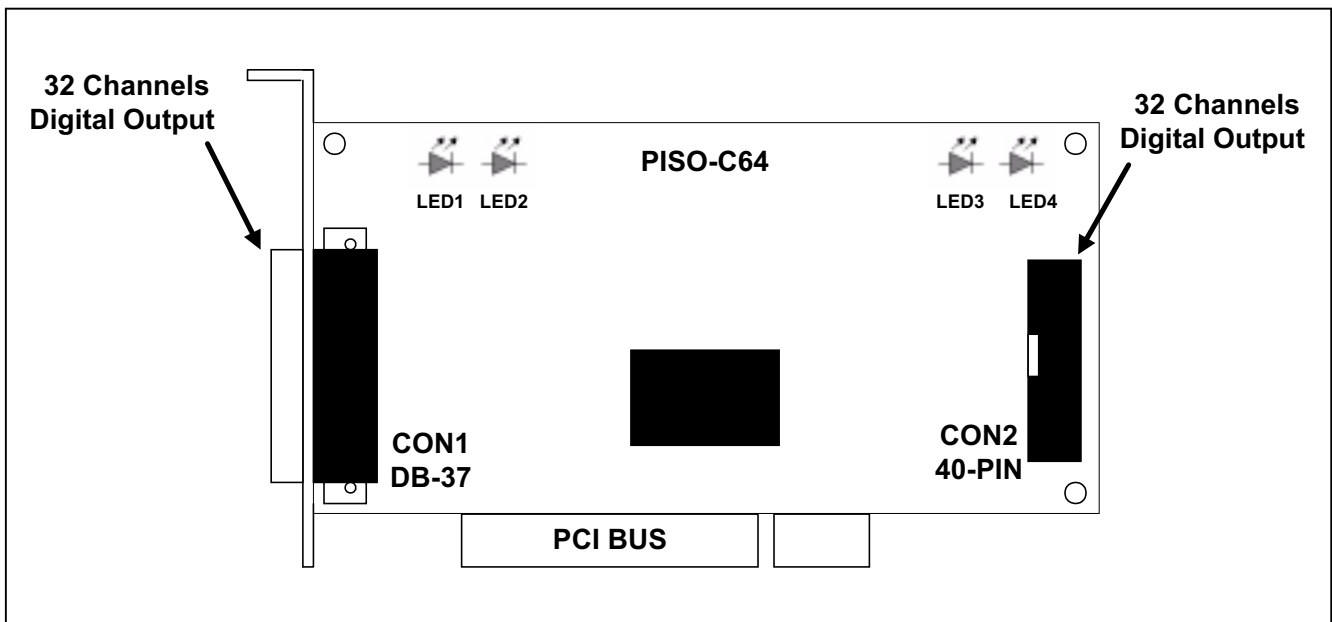


Figure 2-1C. Board layout of PISO-C64

LED 1: power indicator for DO_0 to DO_15

LED 2: power indicator for DO_16 to DO_31

LED 3: power indicator for DO_32 to DO_47

LED 4: power indicator for DO_48 to DO_63

Isolation bank 1: DO_0 to DO_15, Power=CON1_18, Ground=CON1_1

Isolation bank 2: DO_16 to DO_31, Power=CON1_37, Ground=CON1_20

Isolation bank 3: DO_32 to DO_47, Power=CON2_18, Ground=CON2_1

Isolation bank 4: DO_48 to DO_63, Power=CON2_37, Ground=CON2_20

All these four banks are fully isolated from each other.

2.2 Isolated D/I Architecture

The D/I architecture of PISO-P32C32 & PISO-P64 are the same. The users can select internal power or external power to supply photo-couple digital input power. The block diagram of D/I are given as follows:

Configure 1: Internal power supply (Default Setting)

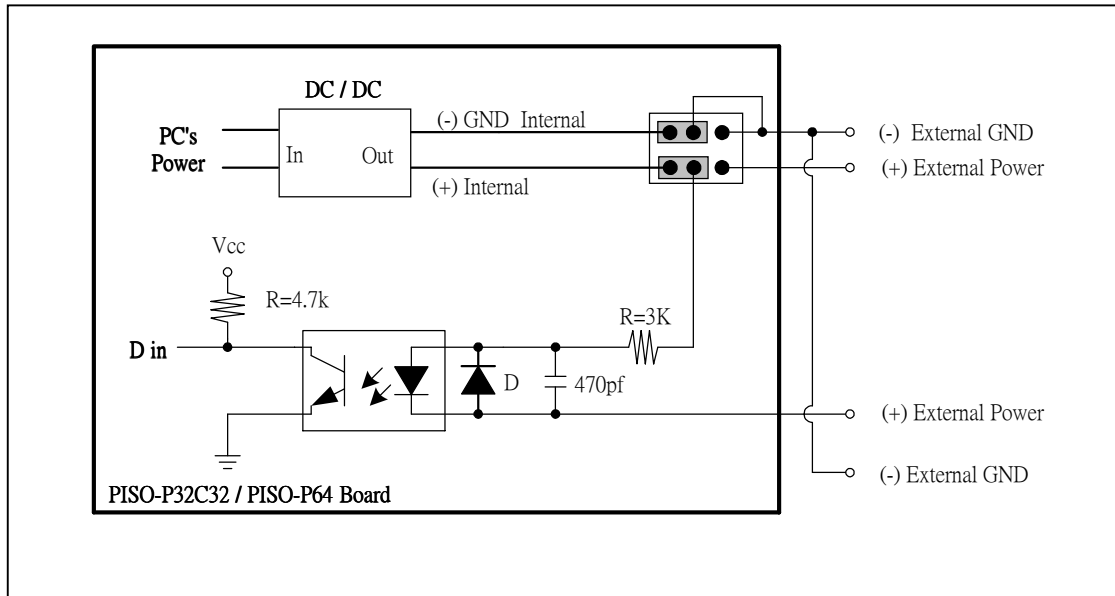


Figure 2-2-1. Isolated D/I Architecture with internal power supply

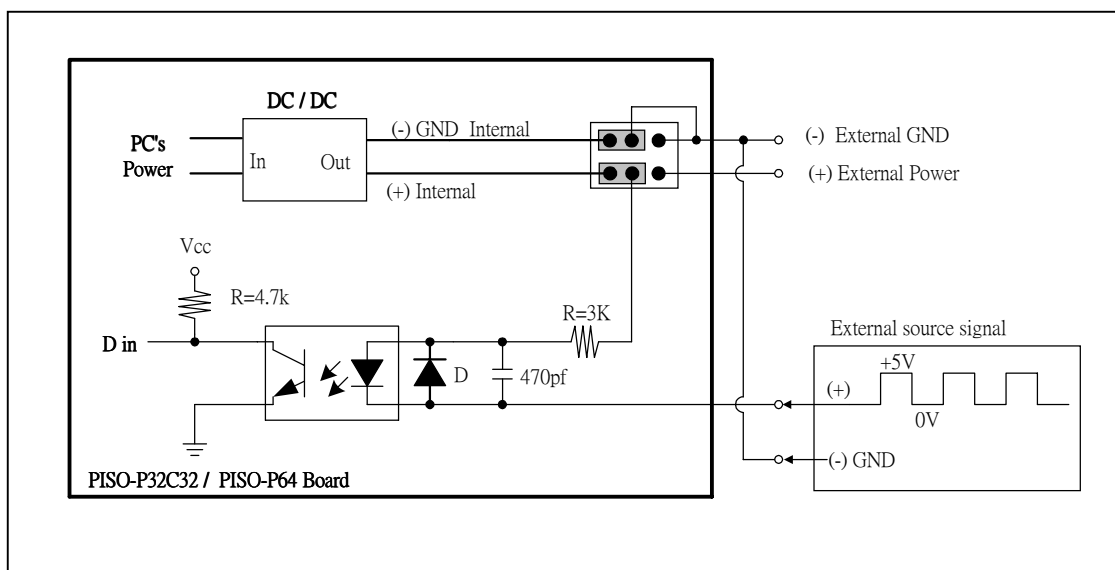


Figure 2-2-2. Typical Applications of D/I with internal power supply

Configure 2: External power supply

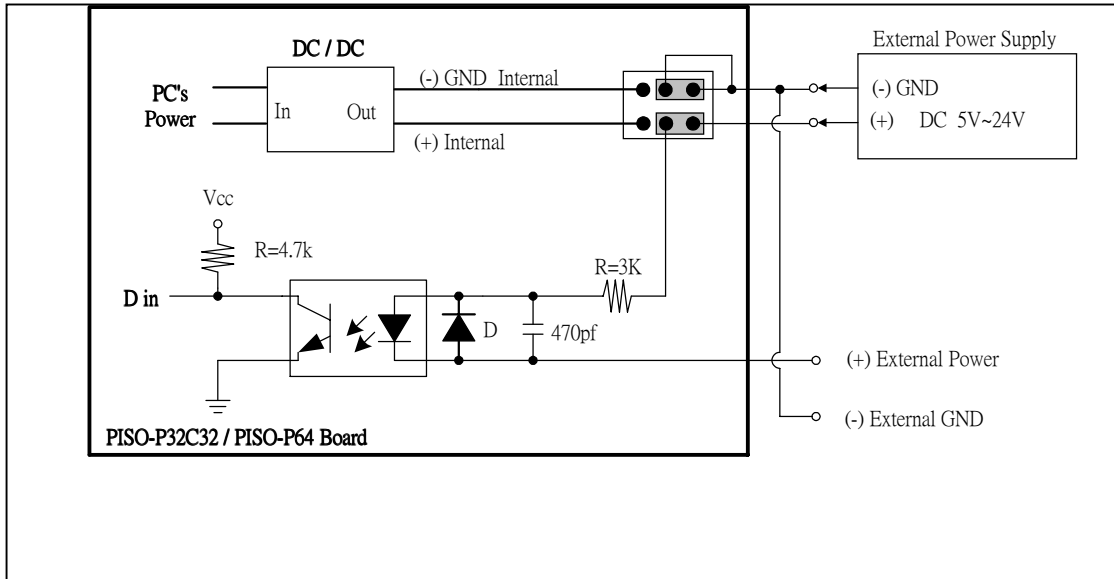


Figure 2-2-3. Isolated D/I Architecture with external power supply

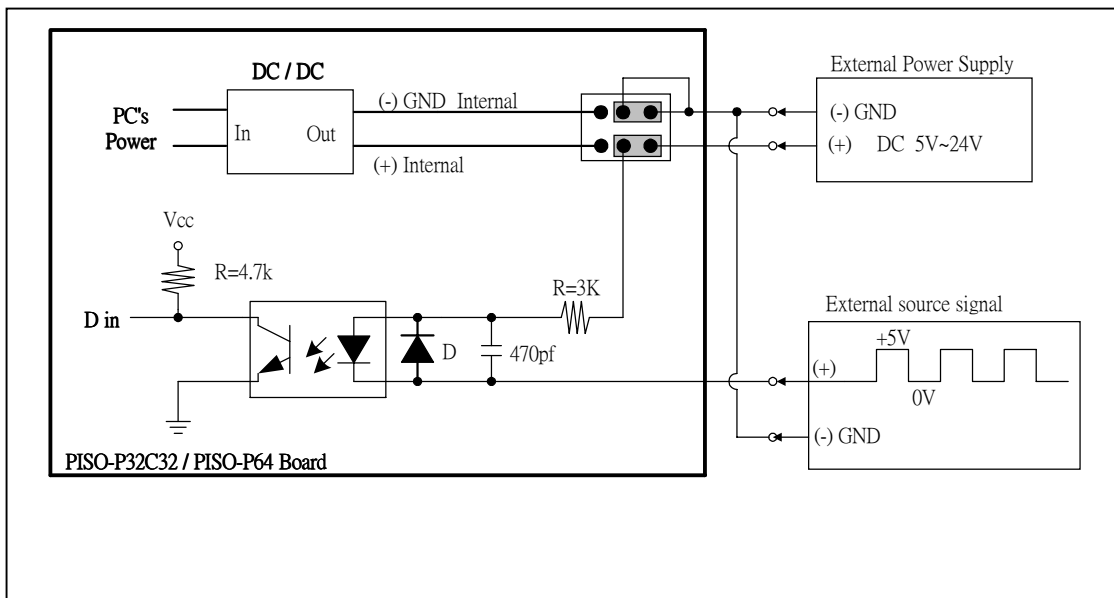
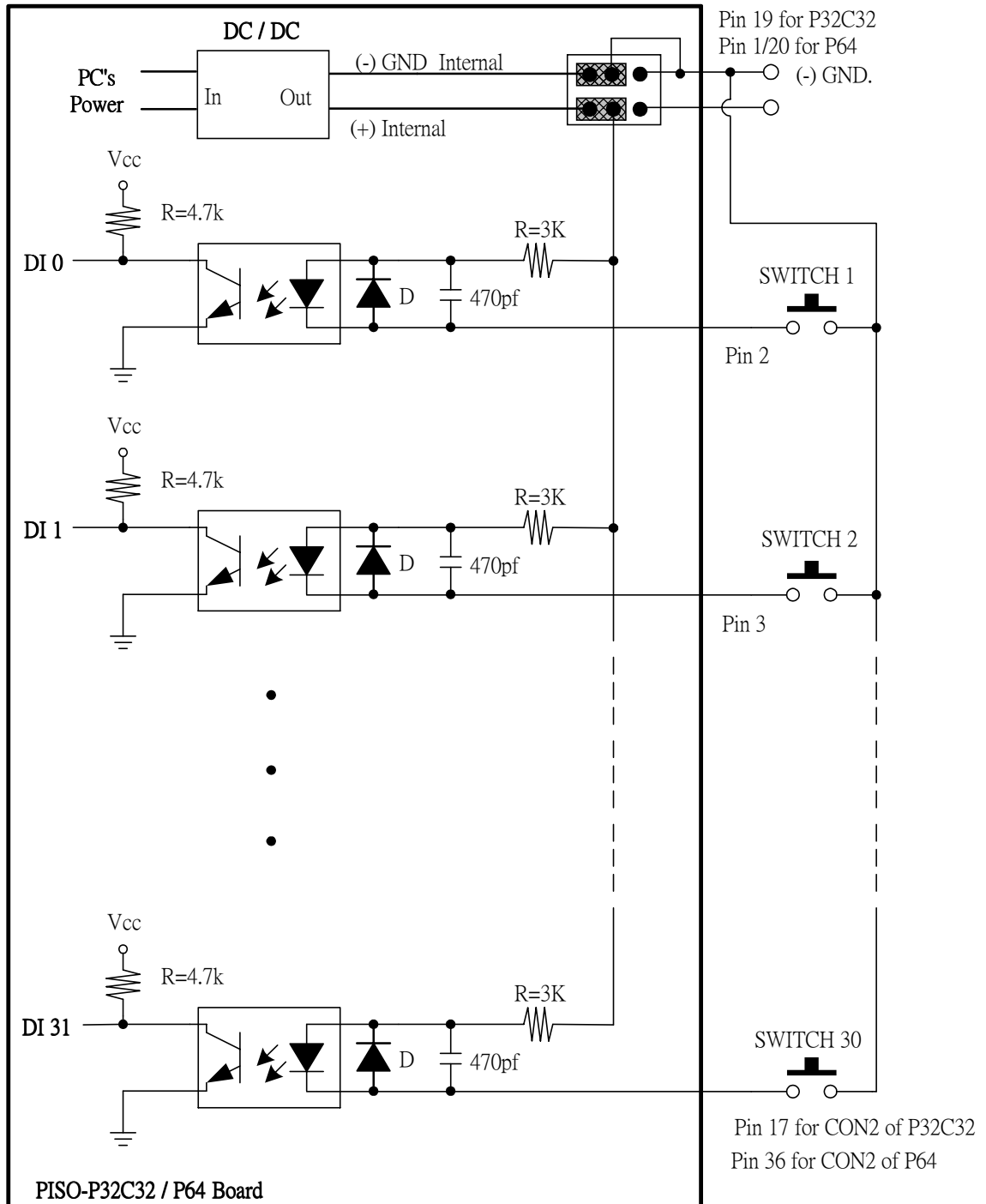
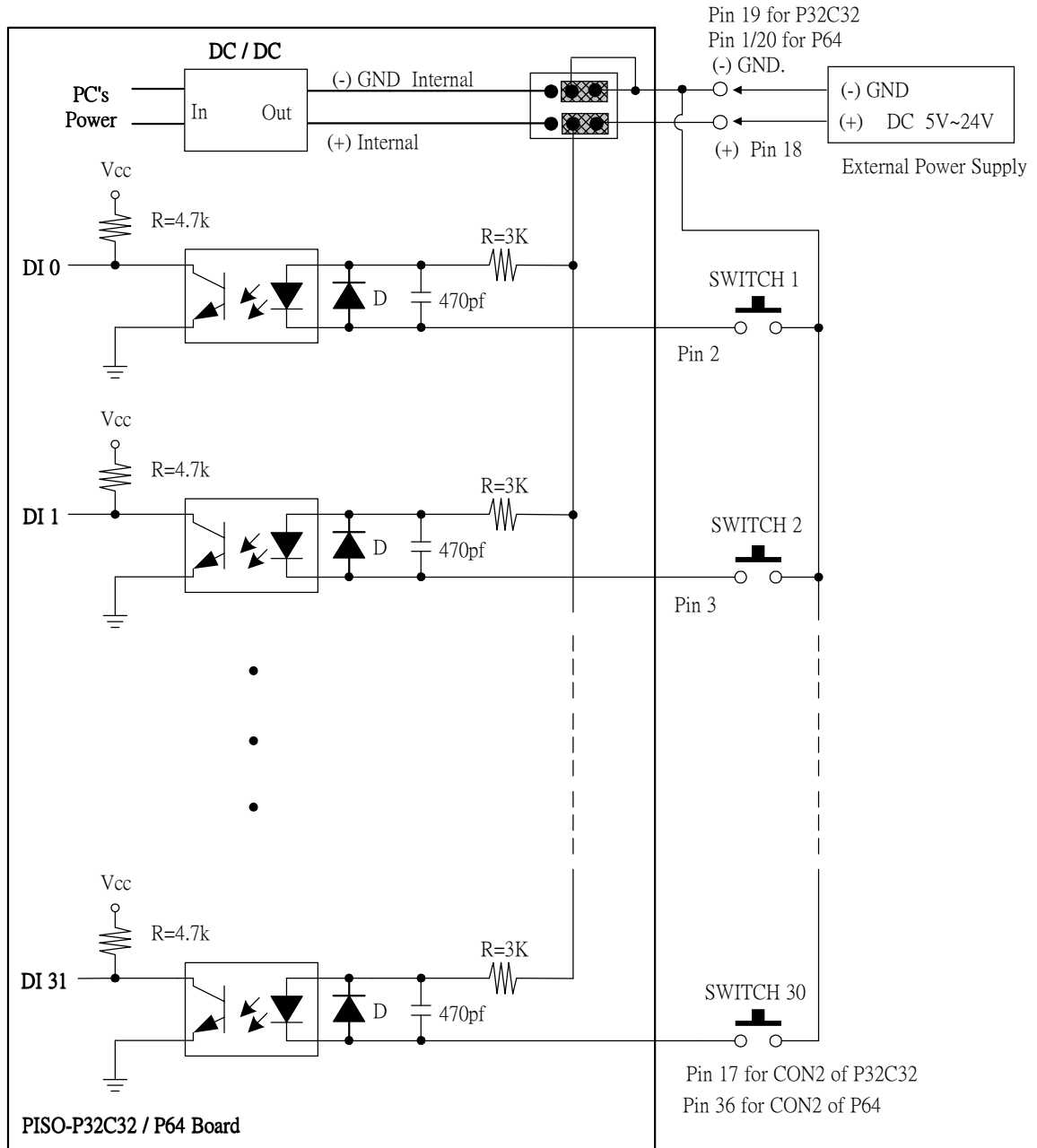


Figure 2-2-4. Typical Applications of D/I with external power supply

2.2.1 Sample D/I Architecture for Internal Power



2.2.2 Sample D/I Architecture for External Power



2.3 Isolated D/O Architecture

The D/O architecture of PISO-P32C32 & PISO-C64 are the same. The block diagram of D/O are given as follows:

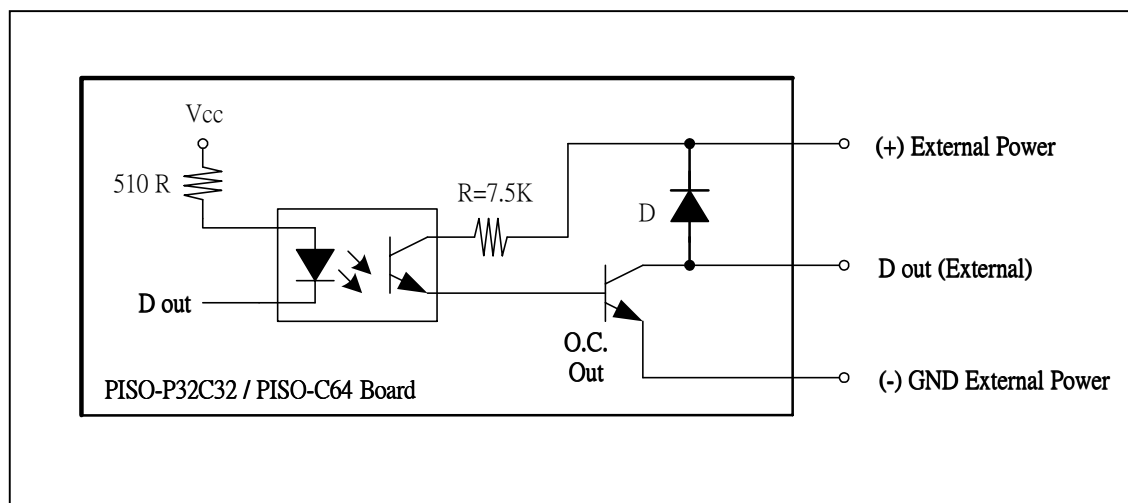


Figure 2-3-1. Isolated D/O Architecture

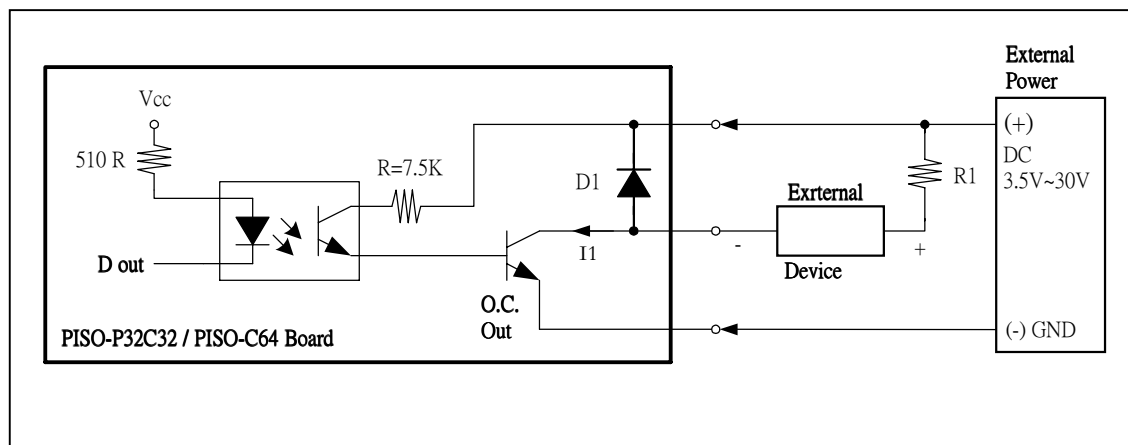
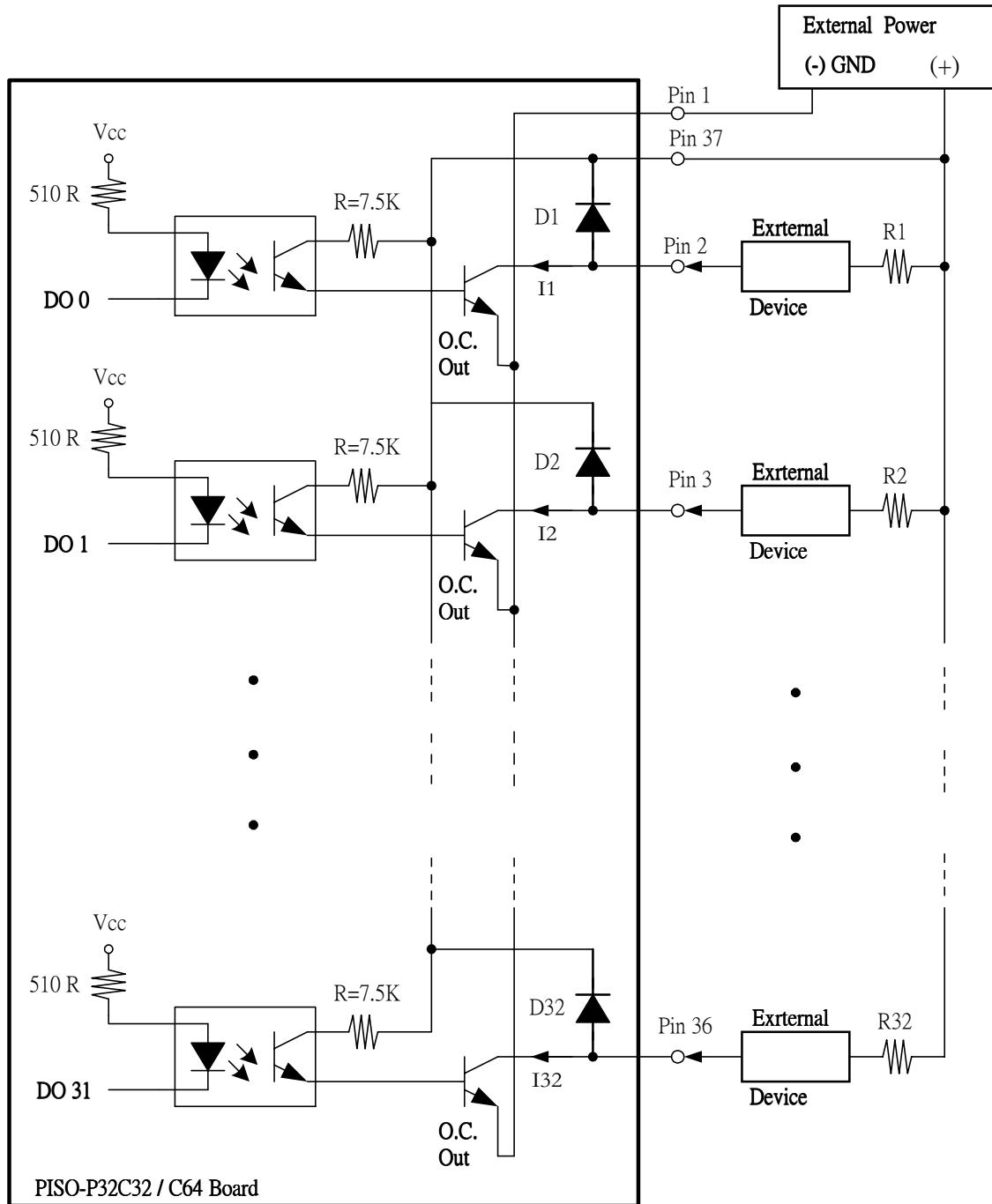


Figure 2-3-2. Typical Applications of D/O

NOTE:

1. The I_1 , I_2 , ... & I_{32} must be < 100 mA
2. The R_1 , R_2 , ... & R_{32} are current-limit resistors. They must be designed to let I_1 , I_2 , ... & $I_{32} < 100$ mA.
3. If the internal resistance of external device is large enough, the R can be omitted.
4. The D_1 , D_2 , ... & D_{31} are common-cathode diodes for switching inductive loads. They can be used as relay driver, hammer driver, lamp driver, display driver, line driver & logic buffers.

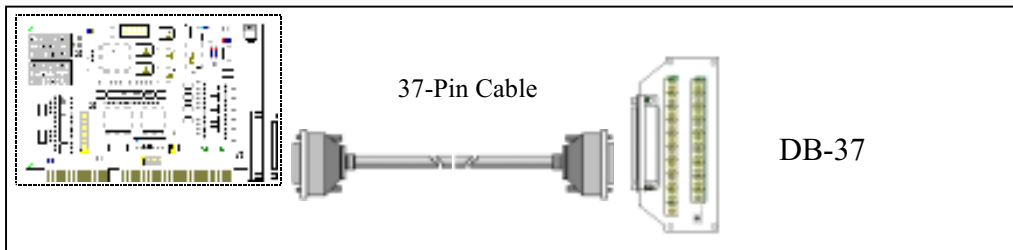
2.3.1 Sample D/O Architecture



2.4 Daughter Boards

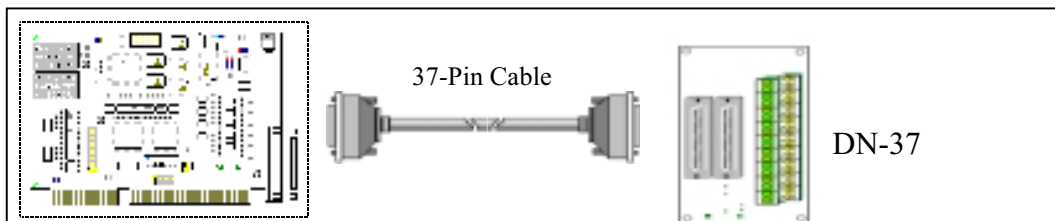
2.4.1 DB-37

The DB-37 is a general purpose daughter board for D-sub 37 pins. It is designed for easy wire connection.



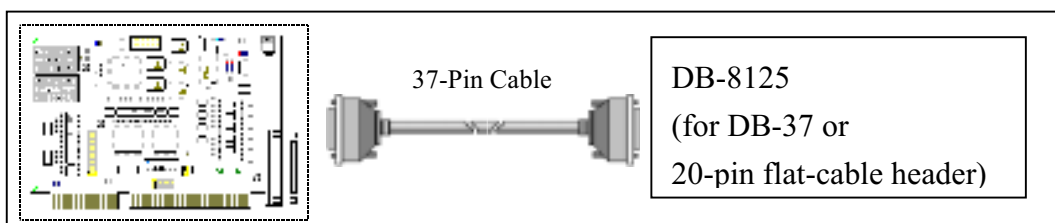
2.4.2 DN-37

The DN-37 is a general purpose daughter board for DB-37 with DIN-Rail Mounting. It is designed for easy wire connection.

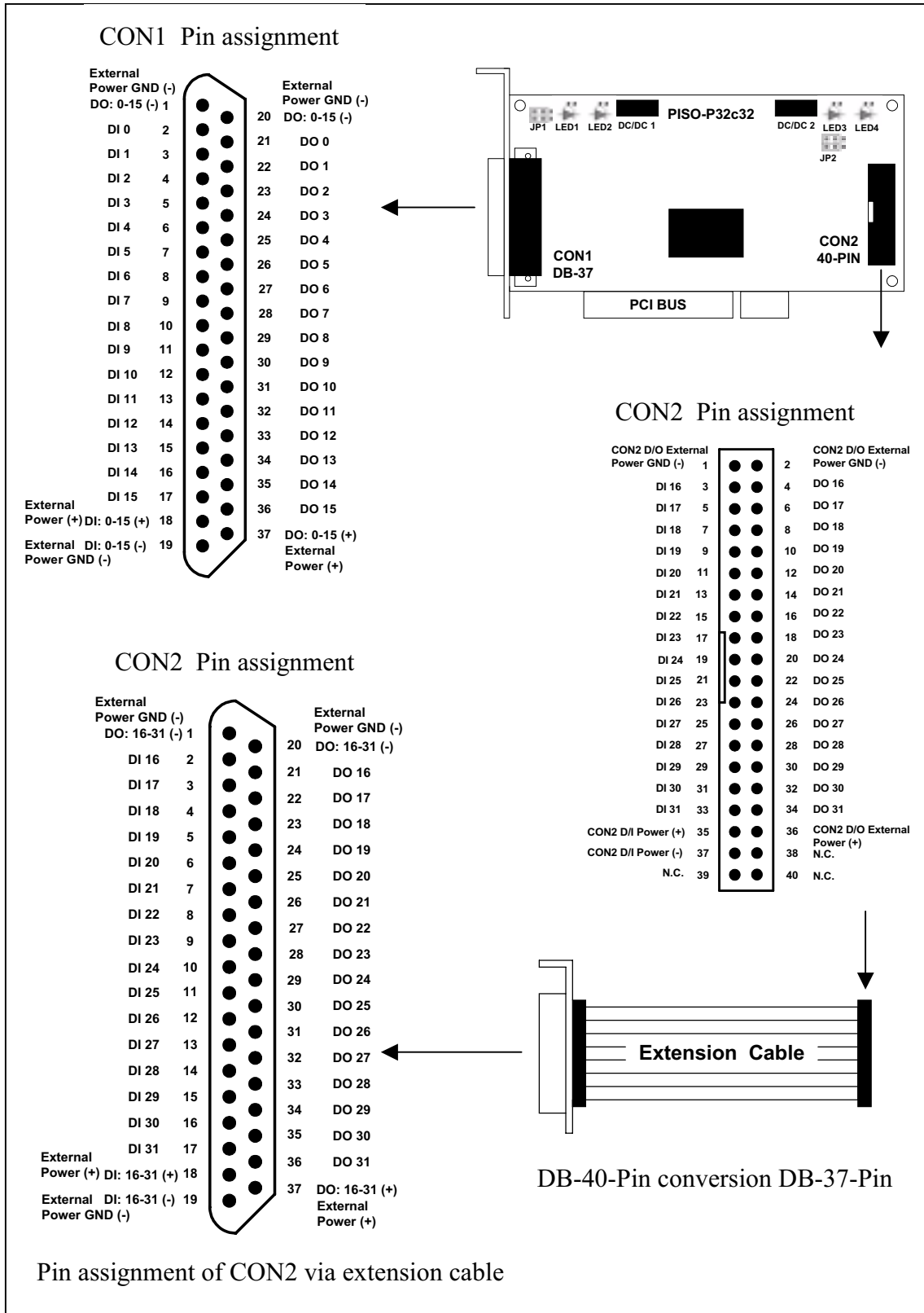


2.4.3 DB-8125

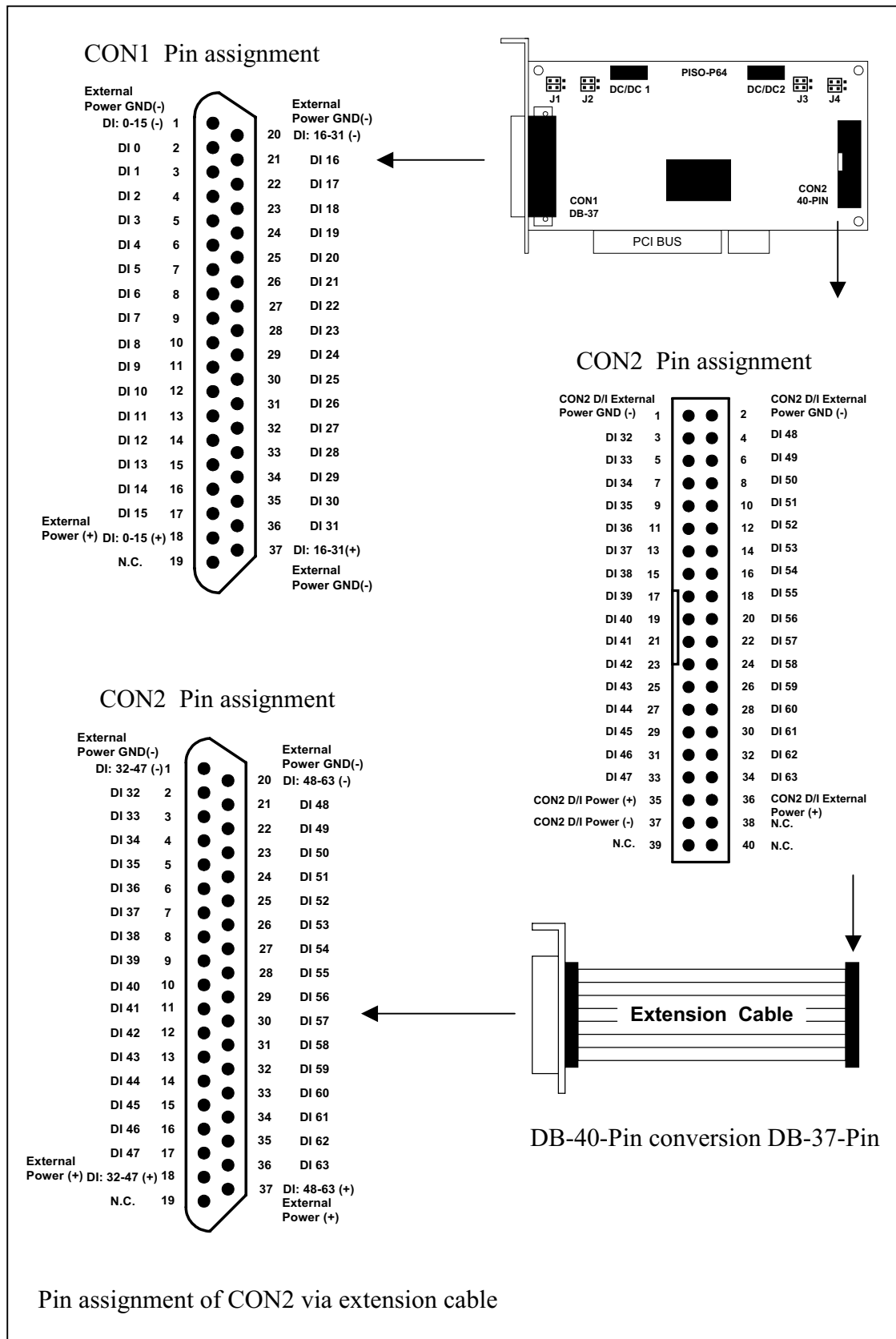
The DB-8125 is a general purpose screw terminal board. It is designed for easy wire connection. There are one DB-37 & two 20-pin flat-cable header in the DB-8125.



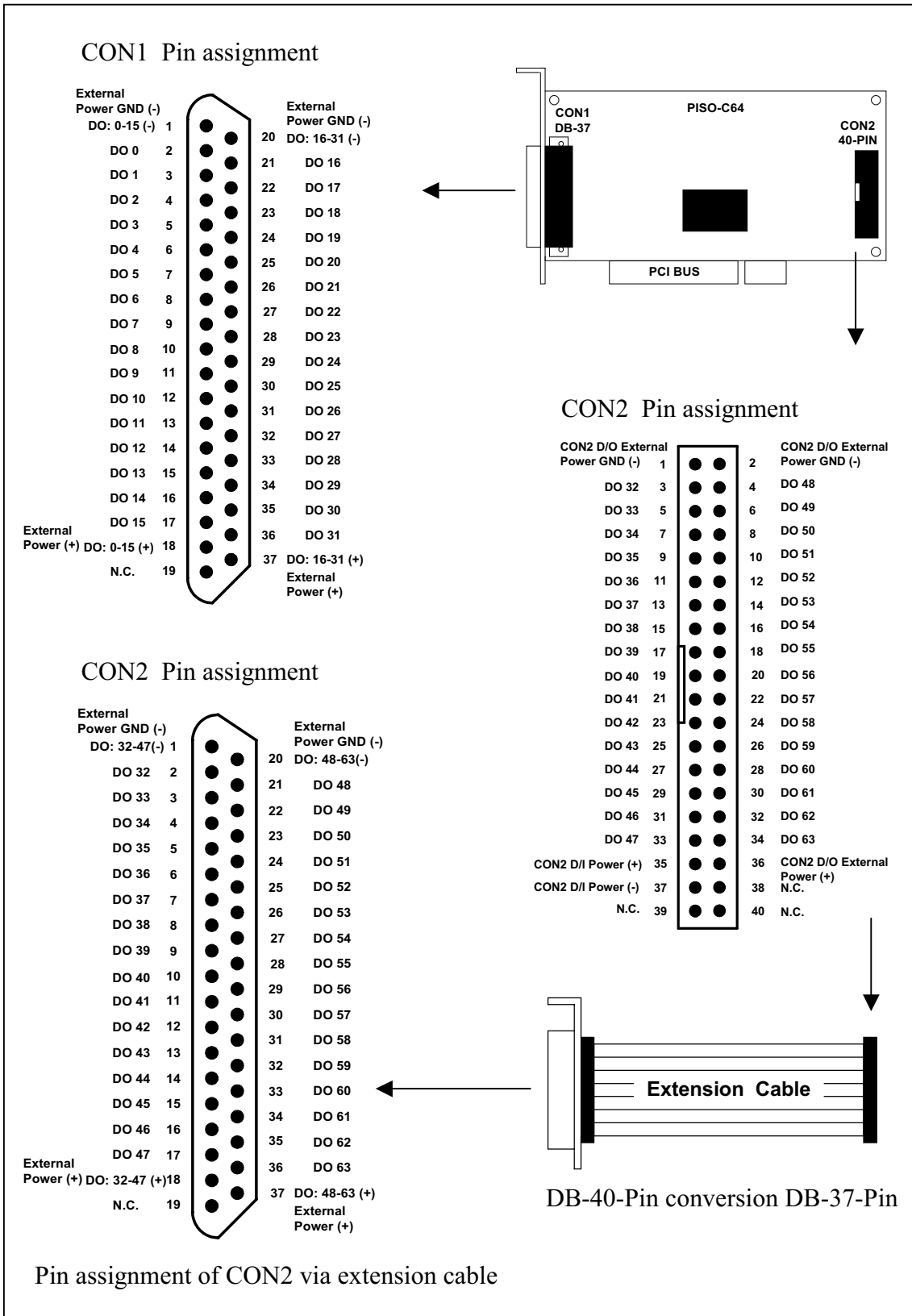
2.5 Pin Assignment of PISO-P32C32



2.6 Pin Assignment of PISO-P64



2.7 Pin Assignment of PISO-C64



3. I/O Control Register

3.1 How to Find the I/O Address

The plug&play BIOS will assign a proper I/O address to every PIO/PISO series card in the power-on stage. The fixed IDs of PIO/PISO series card are given as follows:

- **Vendor ID** = **E159**
- **Device ID** = **0002**

The sub IDs of **PISO-P32C32/P64/C64** are given as follows:

- **Sub-Vendor ID** = **80**
- **Sub-Device ID** = **08**
- **Sub-Aux ID** = **00** → **PISO-C64**
10 → **PISO-P64**
20 → **PISO-P32C32**

We provide all necessary functions as follows:

1. **PIO_DriverInit(&wBoard, wSubVendor, wSubDevice, wSubAux)**
2. **PIO_GetConfigAddressSpace(wBoardNo,*wBase,*wIrq, *wSubVendor, *wSubDevice, *wSubAux, *wSlotBus, *wSlotDevice)**
3. **Show_PIO_PISO(wSubVendor, wSubDevice, wSubAux)**

All functions are defined in PIO.H. Refer to Chapter 4 for more information. The important driver information is given as follows:

1. Resource-allocated information:

- wBase : BASE address mapping in this PC
- wIrq: IRQ channel number allocated in this PC

2. PIO/PISO identification information:

- wSubVendor: subVendor ID of this board
- wSubDevice: subDevice ID of this board
- wSubAux: subAux ID of this board

3. PC's physical slot information:

- wSlotBus: hardware slot ID1 in this PC's slot position
- wSlotDevice: hardware slot ID2 in this PC's slot position

The utility program, **PIO_PISO.EXE**, will detect & show all PIO/PISO cards installed in this PC. Refer to Chapter 5 for more information.

The sub IDs of PIO/PISO series card are given as follows:

PIO/PISO series card	Description	Sub_Vendor	Sub_Device	Sub_AUX
PIO-D144	144 * D/I/O	80	01	00
PIO-D96	96 * D/I/O	80	01	10
PIO-D64	64 * D/I/O	80	01	20
PIO-D56	24* D/I/O + 16*D/I + 16*D/O	80	01	40
PIO-D48	48*D/I/O	80	01	30
PIO-D24	24*D/I/O	80	01	40
PIO-823	Multi-function	80	03	00
PIO-821	Multi-function	80	03	10
PIO-DA16	16*D/A	80	04	00
PIO-DA8	8*D/A	80	04	00
PIO-DA4	4*D/A	80	04	00
PISO-C64	64 * isolated D/O	80	08	00
PISO-P64	64 * isolated D/I	80	08	10
PISO-P32C32	32 + 32	80	08	20
PISO-P8R8	8* isolated D/I + 8 * 220V relay	80	08	30
PISO-P8SSR8AC	8* isolated D/I + 8 * SSR /AC	80	08	30
PISO-P8SSR8DC	8* isolated D/I + 8 * SSR /DC	80	08	30
PISO-730	16*DI+16*D/O + 16* isolated D/I + 16* isolated D/O	80	08	40
PISO-813	32 * isolated A/D	80	0A	00
PISO-DA2	2 * isolated D/A	80	0B	00

3.1.1 PIO_DriverInit

PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux)

- wBoards=0 to N → Number of boards found in this PC
- wSubVendor → SubVendor ID of board to find
- wSubDevice → SubDevice ID of board to find
- wSubAux → SubAux ID of board to find

This function can detect all PIO/PISO series card in the system. It is implemented based on the PCI plug&play mechanism-1. It will find all PIO/PISO series cards installed in this system & save all their resource in the library.

Find all PIO/PISO in this PC

```

/* Step 1:Detect all PIO/PISO series in this PC */
wRetVal=PIO_DriverInit(&wBoards, 0xff, 0xff, 0xff);          /*Find all PIO_PISO*/
printf("\nThrer are %d PIO_PISO Cards in this PC",wBoards);
if (wBoards==0 ) exit(0);

/* Step2: Save resource of all PIO/ISO cards installed in this PC */
printf("\n-----");
for(i=0; i<wBoards; i++)
{
    PIO_GetConfigAddressSpace(i, &wBase, &wIrq, &wSubVendor, &wSubDevice, &wSubAux,
                              &wSlotBus, &wSlotDevice);

    printf("\nCard_%d:wBase=%x,wIrq=%x,subID=[%x,%x,%x],
           SlotID=[%x,%x]",i,wBase,wIrq,wSubVendor,wSubDevice,
           wSubAux,wSlotBus,wSlotDevice);
    printf(" --> ");
    ShowPioPiso(wSubVendor,wSubDevice,wSubAux);
}

```

Find all PISO-P32C32 in this PC

```
/* Step1: Detect all PISO-P32C32 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x20; /* for PISO_P32C32 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor, wSubDevice, wSubAux);
printf("Threr are %d PISO-P32C32 Cards in this PC\n",wBoards);

/* Step2: Save resource of all PISO-P32C32 cards installed in this PC */
for (i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i, &wBase, &wIrq, &wID1, &wID2, &wID3, &wID4, &wID5);
printf("\nCard_%d: wBase=%x, wIrq=%x", i, wBase, wIrq);
wConfigSpace[i][0]=wBaseAddress;          /* save all resource of this card */
wConfigSpace[i][1]=wIrq;                  /* save all resource of this card */
}
```

Find all PISO-P64 in this PC

```
/* Step1: Detect all PISO-P64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x10; /* for PISO_P64 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PISO-P64 Cards in this PC\n",wBoards);

/* Step2: save resource of all PISO-P64 cards installed in this PC */
for (i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i, &wBase, &wIrq, &wID1, &wID2, &wID3, &wID4, &wID5);
printf("\nCard_%d: wBase=%x, wIrq=%x", i, wBase, wIrq);
wConfigSpace[i][0]=wBaseAddress;          /* save all resource of this card */
wConfigSpace[i][1]=wIrq;                  /* save all resource of this card */
}
```

Find all PISO-C64 in this PC

```
/* Step1: Detect all PISO-C64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x00; /* for PISO_P64 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PISO-C64 Cards in this PC\n",wBoards);

/* Step2: save resource of all PISO-C64 cards installed in this PC */
for (i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wID1,&wID2,&wID3,&wID4, &wID5);
printf("\nCard_%d: wBase=%x, wIrq=%x", i, wBase, wIrq);
wConfigSpace[i][0]=wBaseAddress; /* save all resource of this card */
wConfigSpace[i][1]=wIrq; /* save all resource of this card */
}
```

3.1.2 PIO_GetConfigAddressSpace

**PIO_GetConfigAddressSpace(wBoardNo,*wBase,*wIrq, *wSubVendor,
*wSubDevice, *wSubAux, *wSlotBus,*wSlotDevice)**

- wBoardNo=0 to N → totally N+1 boards found by PIO_DriverInit(...)
- wBase → base address of the board control word
- wIrq → allocated IRQ channel number of this board
- wSubVendor → subVendor ID of this board
- wSubDevice → subDevice ID of this board
- wSubAux → subAux ID of this board
- wSlotBus → hardware slot ID1 of this board
- wSlotDevice → hardware slot ID2 of this board

The user can use this function to save resource of all PIO/PISO cards installed in this system. Then the application program can control all functions of PIO/PISO series card directly.

Find the configure address space of PIO_P32C32

```

/* Step1: Detect all PIO-P32C32 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x20; /* for PIO_P32C32 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PIO-P32C32 Cards in this PC\n",wBoards);

/* Step2: Save resource of all PIO-P32C32 cards installed in this PC */
for (i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&t1,&t2,&t3,&t4,&t5);
printf("\nCard_%d: wBase=%x, wIrq=%x", i,wBase,wIrq);
wConfigSpace[i][0]=wBaseAddress; /* save all resource of this card */
wConfigSpace[i][1]=wIrq; /* save all resource of this card */
}

/* Step3: Control the PIO-P32C32 directly */
wBase=wConfigSpace[0][0]; /* get base address the card_0 */
outport(wBase,1); /* enable all D/I/O operation of card_0 */

wBase=wConfigSpace[1][0]; /* get base address the card_1 */
outport(wBase,1); /* enable all D/I/O operation of card_1 */

```

Find the configure address space of PIO_P64

```

/* Step1: Detect all PIO-P64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x10;    /* for PIO_P64 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PIO-P64 Cards in this PC\n",wBoards);

/* Step2: Save resource of all PIO-P64 cards installed in this PC */
for (i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&t1,&t2,&t3,&t4,&t5);
printf("\nCard_%d: wBase=%x, wIrq=%x", i,wBase,wIrq);
wConfigSpace[i][0]=wBaseAddress;                /* save all resource of this card */
wConfigSpace[i][1]=wIrq;                        /* save all resource of this card */
}

/* Step3: Control the PIO-P64 directly */
wBase=wConfigSpace[0][0];                        /* get base address the card_0 */
outport(wBase,1);                               /* enable all D/I/O operation of card_0 */

wBase=wConfigSpace[1][0];                        /* get base address the card_1 */
outport(wBase,1);                               /* enable all D/I/O operation of card_1 */

```

Find the configure address space of PIO_C64

```
/* Step1: Detect all PIO-C64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x00; /* for PIO_C64 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PIO-C64 Cards in this PC\n",wBoards);

/* Step2: Save resource of all PIO-C64 cards installed in this PC */
for (i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&t1,&t2,&t3,&t4,&t5);
printf("\nCard_%d: wBase=%x, wIrq=%x", i,wBase,wIrq);
wConfigSpace[i][0]=wBaseAddress; /* save all resource of this card */
wConfigSpace[i][1]=wIrq; /* save all resource of this card */
}

/* Step3: Control the PIO-C64 directly */
wBase=wConfigSpace[0][0]; /* get base address the card_0 */
outport(wBase,1); /* enable all D/I/O operation of card_0 */

wBase=wConfigSpace[1][0]; /* get base address the card_1 */
outport(wBase,1); /* enable all D/I/O operation of card_1 */
```

3.1.3 Show_PIO_PISO

Show_PIO_PISO(wSubVendor, wSubDevice, wSubAux)

- wSubVendor → subVendor ID of board to find
- wSubDevice → subDevice ID of board to find
- wSubAux → subAux ID of board to find

This function will show a text string for this special subIDs. This text string is the same as that defined in PIO.H

The demo program is given as follows:

```
wRetVal=PIO_DriverInit(&wBoards,0xff,0xff,0xff); /* find all PIO_PISO series card*/
printf("\nThrer are %d PIO_PISO Cards in this PC",wBoards);
if (wBoards==0 ) exit(0);

printf("\n-----");
for(i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,
    &wSubDevice,&wSubAux,&wSlotBus,&wSlotDevice);

printf("\nCard_ %d:wBase=%x,wIrq=%x,subID=[%x,%x,%x],
    SlotID=[%x,%x]",i,wBase,wIrq,wSubVendor,wSubDevice,
    wSubAux,wSlotBus,wSlotDevice);
printf(" --> ");
ShowPioPiso(wSubVendor,wSubDevice,wSubAux);
}
```

3.2 The Assignment of I/O Address

The Plug&Play BIOS will assign the proper I/O address to PIO/PISO series card. If there is only one PIO/PISO board, the user can identify the board as card_0. If there are two PIO/PISO boards in the system, the user will be very difficult to identify which board is card_0 ? The software driver can support 16 boards max. Therefore the user can install 16 boards of PIO/PSIO series in one PC system. How to find the card_0 & card_1 ?

It is difficult to find the card NO. The simplest way to identify which card is card_0 is to use wSlotBus & wSlotDevice as following:

1. Remove all PISO-P32C32/P64/C64 from this PC
2. Install one PISO-P32C32/P64/C64 into the PC's PCI_slot1, run PIO_PISO.EXE & record the wSlotBus1 & wSlotDevice1
3. Remove all PISO-P32C32/P64/C64 from this PC
4. Install one PISO-P32C32/P64/C64 into the PC's PCI_slot2, run PIO_PISO.EXE & record the wSlotBus2 & wSlotDevice2
5. Repeat (3) & (4) for all PCI_slot?, record all wSlotBus? & wSlotDevice?

The records may be as following:

PC's PCI slot	WSlotBus	wSlotDevice
Slot 1	0	0x07
Slot 2	0	0x08
Slot 3	0	0x09
Slot 4	0	0x0A
PCI-BRIDGE		
Slot 5	1	0x0A
Slot 6	1	0x08
Slot 7	1	0x09
Slot 8	1	0x07

The above procedure will record all wSlotBus? & wSlotDevice? in this PC. These values will be mapped to this PC's physical slot. This mapping will not be changed for any PIO/PISO cards. So it can be used to identify the specified PIO/PISO card as following:

Step1: Record all wSlotBus? & wSlotDevice?

Step2: Use PIO_GetConfigAddressSpace(...) to get the specified card's wSlotBus & wSlotDevice

Step3: The user can identify the specified PIO/PISO card if he compare the wSlotBus & wSlotDevice in step2 to step1.

3.3 Enable I/O Operation

When the PC is first power-on, all operation of D/I/O port are disable. The enable/disable of D/I/O is controlled by the RESET\ signal. The power-on states are given as follows:

- All D/I/O operations are disable
- All D/O latch register are clear

The user has to perform some initialization before using these D/I/O. The recommended steps are given as follows:

Step 1: enable all D/I/O operation.

Step 2: read from D/I or write to D/O

Refer to DEMO1.C for demo program.

3.4 The I/O Address Map

The I/O address of PIO/PISO series card is automatically assigned by the main board ROM BIOS. The I/O address can also be re-assigned by user. **It is strongly recommended not to change the I/O address by user. The Plug&Play BIOS will assign proper I/O address to each PIO/PISO series card very well.**

3.4.1 PISO-P32C32 I/O Mapping

The I/O address mapping of PISO-P32C32 are given as follows:

Address	Read	Write
Wbase+0	RESET\ control register	Same
Wbase+2	Aux control register	Same
Wbase+3	Aux data register	Same
Wbase+5	INT mask control register	Same
Wbase+7	Aux pin status register	same
Wbase+0x2a	INT polarity control register	same
Wbase+0xc0	Read data from DI 0 ~ DI 7	Write data to DO 0 to DO 7
Wbase+0xc4	Read data from DI 8 ~ DI 15	Write data to DO 8 to DO 15
Wbase+0xc8	Read data from DI 16 ~ DI 23	Write data to DO 16 to DO 23
Wbase+0xcc	Read data from DI 24 ~ DI 31	Write data to DO 24 to DO 31

Note. Refer to Sec. 3.1 for more information about wBase.

```

outportb(wBase+0xc0,Val);          /* write to D/O 0~7 */
outportb(wBase+0xc4,Val);          /* write to D/O 8~15 */
outportb(wBase+0xc8,Val);          /* write to D/O 16~23 */
outportb(wBase+0xcc,Val);          /* write to D/O 24~31 */

Val=inportb(wBase+0xc0);           /* read from D/I 0~7 */
Val=inportb(wBase+0xc4);           /* read from D/I 8~15 */
Val=inportb(wBase+0xc8);           /* read from D/I 16~23 */
Val=inportb(wBase+0xcc);           /* read from D/I 24~31 */

```

3.4.2 PISO-P64 I/O Mapping

The I/O address mapping of PISO-P64 are given as follows:

Address	Read	Write
wBase+0	RESET\ control register	Same
wBase+2	Aux control register	Same
wBase+3	Aux data register	Same
WBase+5	INT mask control register	Same
Wbase+7	Aux pin status register	same
Wbase+0x2a	INT polarity control register	same
Wbase+0xc0	Read data from DI_0 ~ DI_7	reserved
Wbase+0xc4	Read data from DI_8 ~ DI_15	reserved
Wbase+0xc8	Read data from DI_16 ~ DI_23	reserved
Wbase+0xcc	Read data from DI_24 ~ DI_31	reserved
WBase+0xd0	Read data from DI_32 ~ DI_39	reserved
WBase+0xd4	Read data from DI_40 ~ DI_47	reserved
WBase+0xd8	Read data from DI_48 ~ DI_55	reserved
WBase+0xdc	Read data from DI_56 ~ DI_63	reserved

Note. Refer to Sec. 3.1 for more information about wBase.

```
Val=inportb(wBase+0xc0);          /* read from D/I 0~7 */
Val=inportb(wBase+0xc4);          /* read from D/I 8~15 */
Val=inportb(wBase+0xc8);          /* read from D/I 16~23 */
Val=inportb(wBase+0xcc);          /* read from D/I 24~31 */
```

```
Val=inportb(wBase+0xd0);          /* read from D/I 32~39 */
Val=inportb(wBase+0xd4);          /* read from D/I 40~47 */
Val=inportb(wBase+0xd8);          /* read from D/I 48~55 */
Val=inportb(wBase+0xdc);          /* read from D/I 56~63 */
```

3.4.3 PISO-C64 I/O Mapping

The I/O address mapping of PISO-C64 are given as follows:

Address	Read	Write
wBase+0	RESET\ control register	Same
wBase+2	Aux control register	Same
wBase+3	Aux data register	Same
wBase+5	INT mask control register	Same
wBase+7	Aux pin status register	same
wBase+0x2a	INT polarity control register	same
wBase+0xc0	Reserved	Write data to DO_0 to DO_7
wBase+0xc4	Reserved	Write data to DO_8 to DO_15
wBase+0xc8	Reserved	Write data to DO_16 to DO_23
wBase+0xcc	Reserved	Write data to DO_24 to DO_31
wBase+0xd0	Reserved	Write data to DO_32 to DO_39
wBase+0xd4	Reserved	Write data to DO_40 to DO_47
wBase+0xd8	Reserved	Write data to DO_48 to DO_55
wBase+0xdc	Reserved	Write data to DO_56 to DO_63

Note. Refer to Sec. 3.1 for more information about wBase.

```

outportb(wBase+0xc0,Val);          /* write to D/O 0~7 */
outportb(wBase+0xc4,Val);          /* write to D/O 8~15 */
outportb(wBase+0xc8,Val);          /* write to D/O 16~23 */
outportb(wBase+0xcc,Val);          /* write to D/O 24~31 */

```

```

outportb(wBase+0xd0,Val);          /* write to D/O 32~39 */
outportb(wBase+0xd4,Val);          /* write to D/O 40~47 */
outportb(wBase+0xd8,Val);          /* write to D/O 48~55 */
outportb(wBase+0xdc,Val);          /* write to D/O 56~63 */

```

3.4.4 RESET\ Control Register

(Read/Write): wBase+0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	RESET\

Note. Refer to Sec. 3.1 for more information about wBase.

When the PC is first power-on, the RESET\ signal is in Low-state. **This will disable all D/I/O operations.** The user has to set the RESET\ signal to High-state before any D/I/O command.

```
outportb(wBase,1);    /* RESET\ = High → all D/I/O are enable now */
```

```
outportb(wBase,0);    /* RESET\ = Low → all D/I/O are disable now */
```

3.4.5 AUX Control Register

(Read/Write): wBase+2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

Note. Refer to Sec. 3.1 for more information about wBase.

Aux?=0 → this Aux is used as a D/I

Aux?=1 → this Aux is used as a D/O

When the PC is first power-on, All Aux? signal are in Low-state. All Aux? are designed as D/I for all PIO/PISO series. Please set all Aux? in D/I state.

3.4.6 AUX data Register

(Read/Write): wBase+3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

Note. Refer to Sec. 3.1 for more information about wBase.

When the Aux? is used as D/O, the output state is controlled by this register. This register is designed for feature extension, so don't control this register now.

3.4.7 INT Mask Control Register

(Read/Write): wBase+5

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0

Note. Refer to Sec. 3.1 for more information about wBase.

This register is designed for feature extension, so don't control this register now.

3.4.8 Aux Status Register

(Read/Write): wBase+7

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

Note. Refer to Sec. 3.1 for more information about wBase.

Aux0-3=reserved, aux4-7=Aux-ID.

4. The applications of Digital I/O

4.1 The example of PISO-P32C32

- The circuit diagram of D/O of PISO-P32C32 is given as follows:

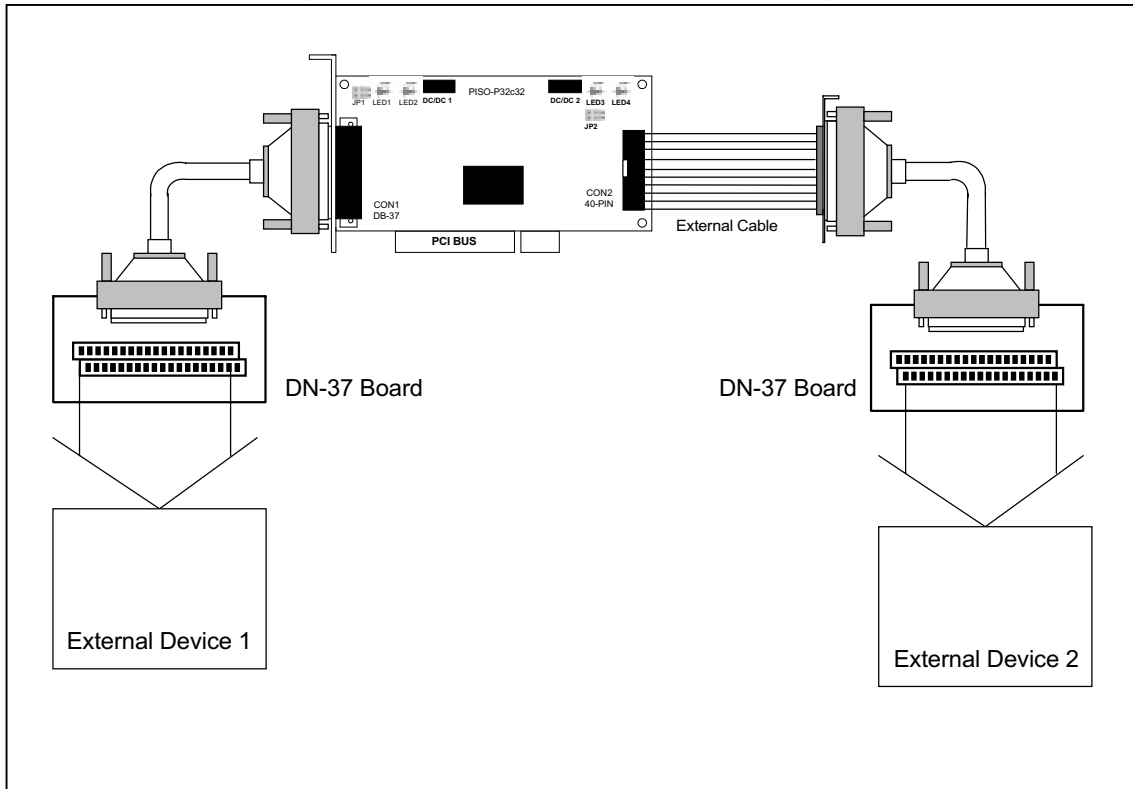
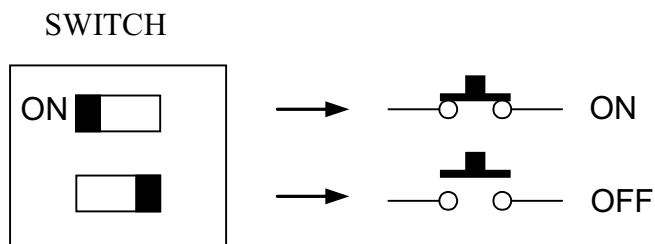


Figure 4-1-1. The example of digital inputs/outputs for PISO-P32C32

- The circuit diagram of external device 1: Refer to Figure 4-1-2
- The circuit diagram of external device 2: Refer to Figure 4-1-3



- The circuit diagram of external device 1 is given as follows:

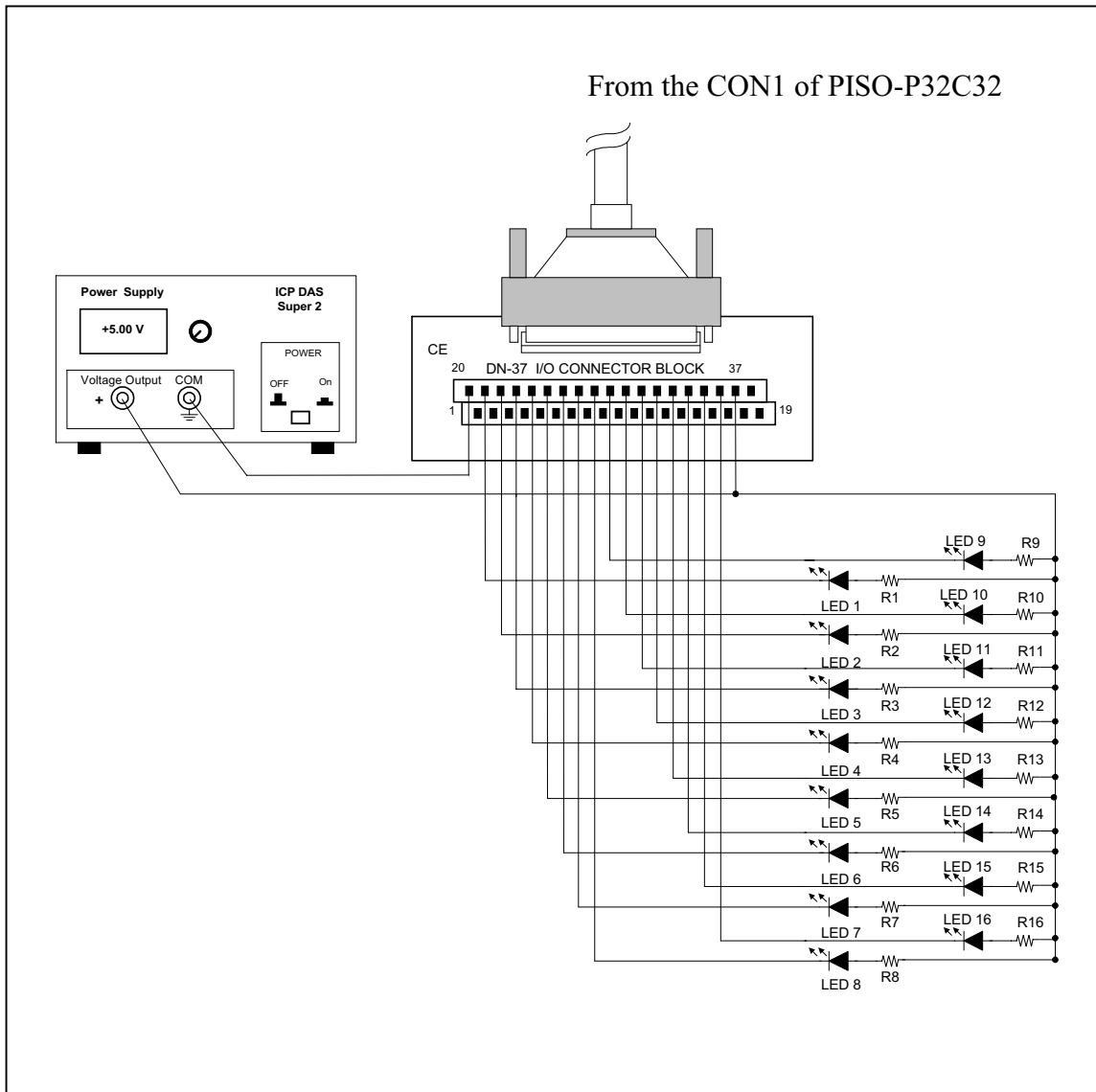


Figure 4-1-2. The circuit diagram of external device 1 for the digital outputs of PISO-P32C32

- The resistor of R1~R16 are 330 ohm.
- The LED 1~LED 16 are light emitting diode.
- The pin-1/20 are the GND signal of DI_0~DI_15 / DO_0~DO_15.
- The pin-18/37 are the voltage (+) signal of DI_0~DI_15 / DO_0~DO_15 (input DC +5V~+24V).

- The circuit diagram of external device 2 is given as follows:

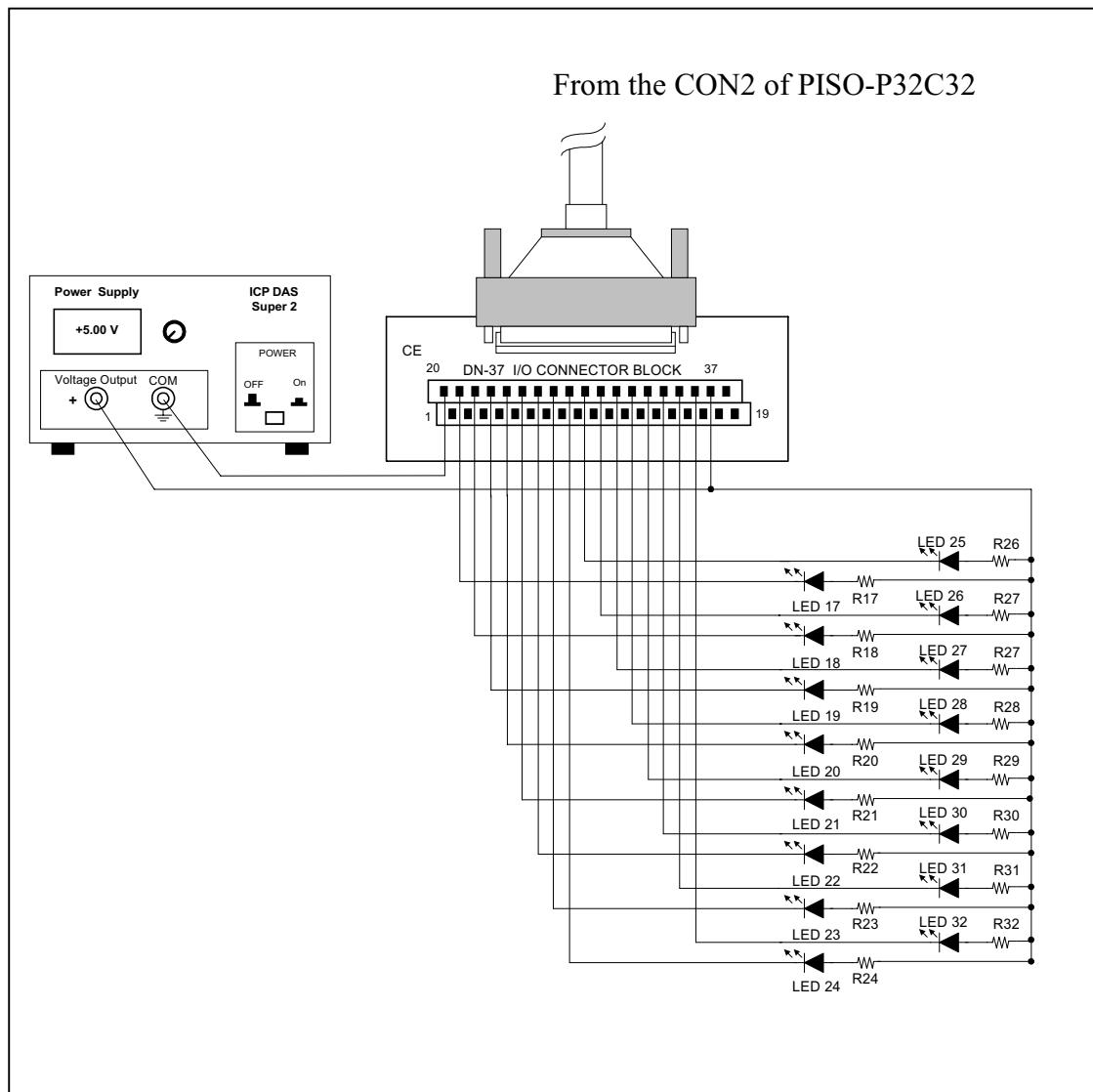


Figure 4-1-3. The circuit diagram of external device 2 for the digital outputs of PISO-P32C32

- The resistor of R17~R32 are 330 ohm.
- The LED 17~LED 32 are light emitting diode.
- The pin-1/20 are the GND signal of DI_16~DI_31 / DO_16~DO_31.
- The pin-18/37 are the voltage (+) signal of DI_16~DI_31 / DO_16~DO_31 (input DC 5V~24V).

- The circuit diagram of D/I of PISO-P32C32 is given as follows:

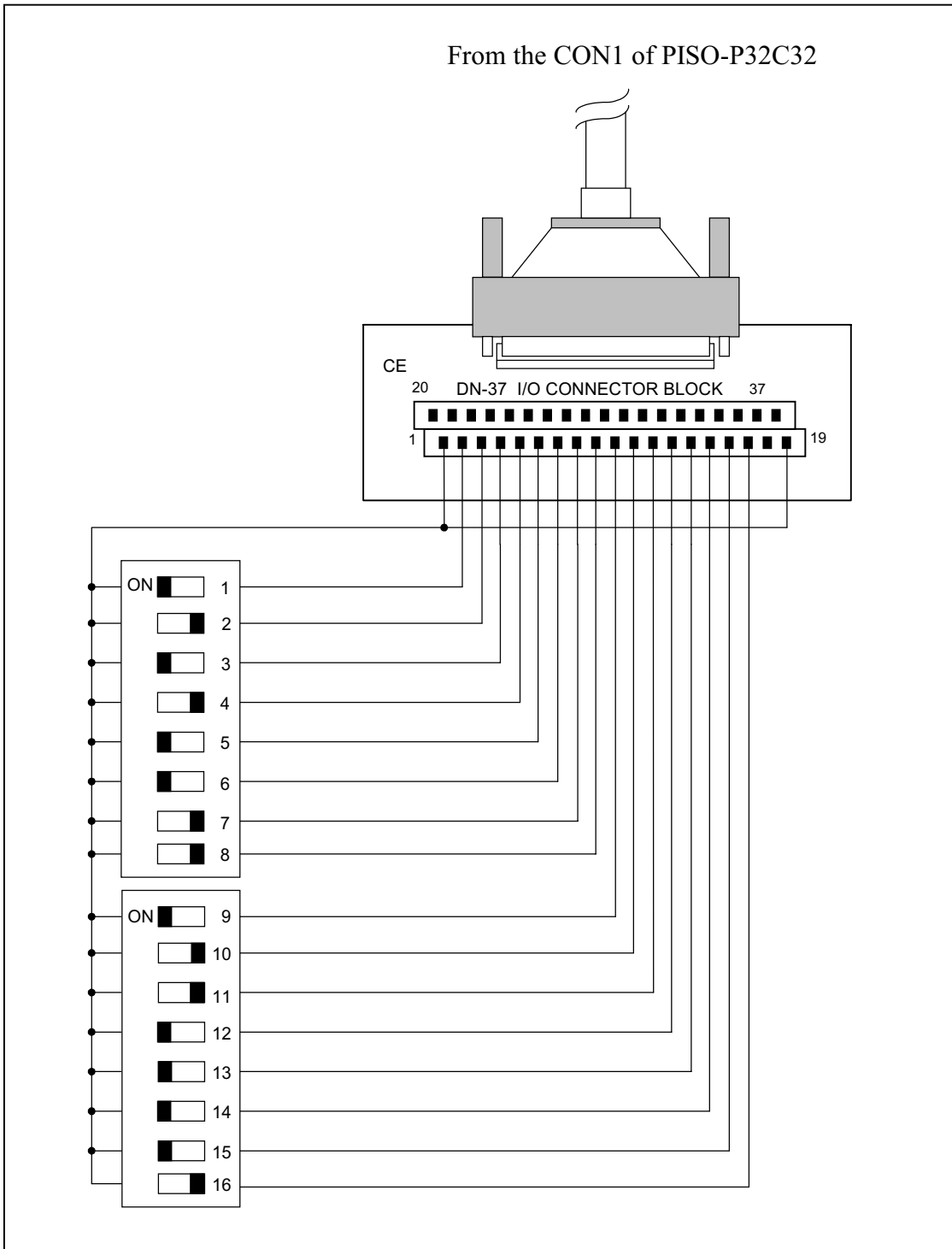


Figure 4-1-4. The circuit diagram of external device 1 for the D/I of PISO-P32C32

- The D/I of CON1 of PISO-P32C32 select **internal power**.
- The pin-19 are the GND signal of DI_0~DI_15.
- The pin-18 are the voltage (+) signal of DI_0~DI_15.

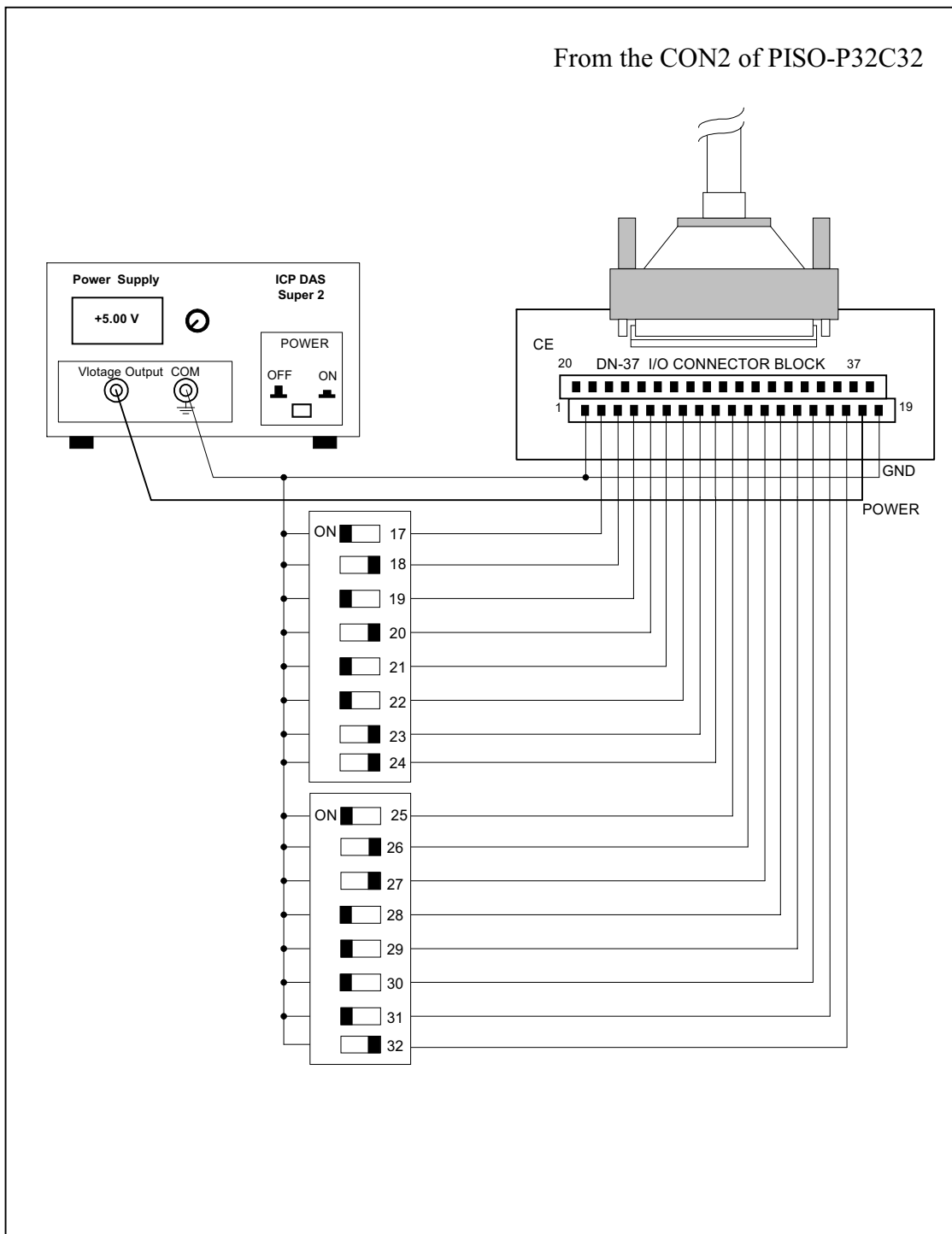


Figure 4-1-5. The circuit diagram of external device 2 for the D/I of PISO-P32C32.

- The D/I of CON1 of PISO-P32C32 select **external power**.
- The pin-19 are the GND signal of DI_0~DI_15.
- The pin-18 are the voltage (+) signal of DI_0~DI_15 (input DC +5V~+24V).

- The test circuit diagram of D/I/O of PISO-P32C32 is given as follows:

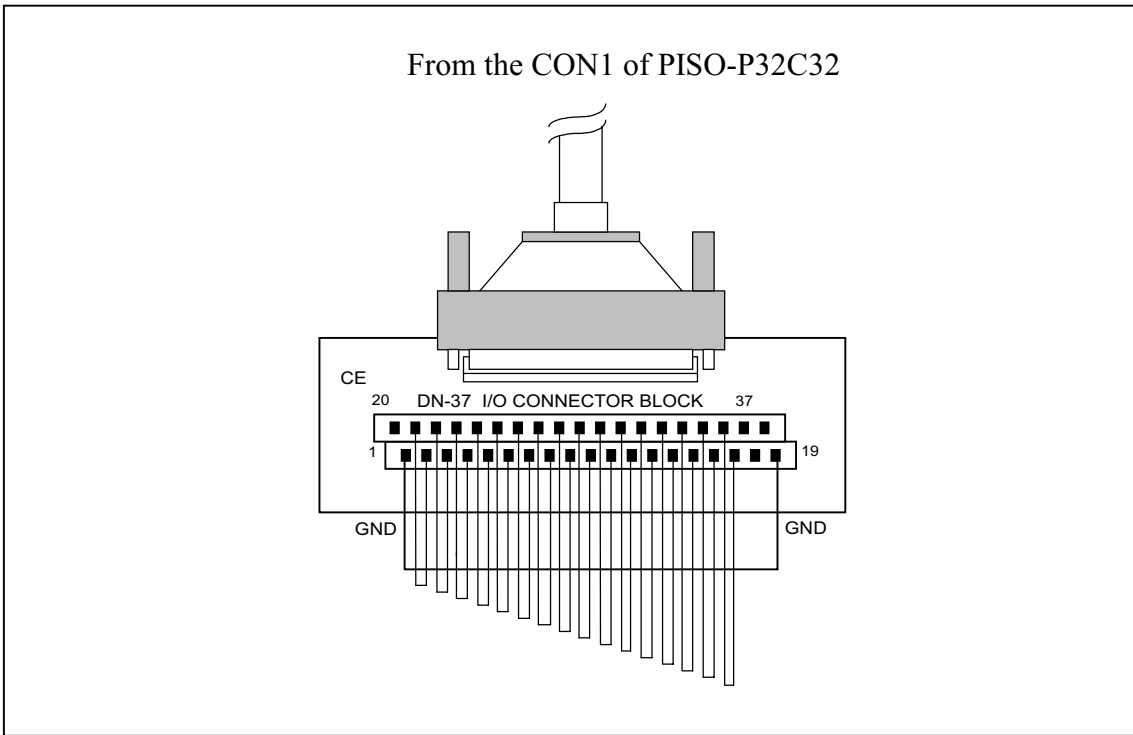


Figure 4-1-4. The D/I/O of CON1 of PISO-P32C32

- The D/I of CON2 of PISO-P32C32 for internal power.

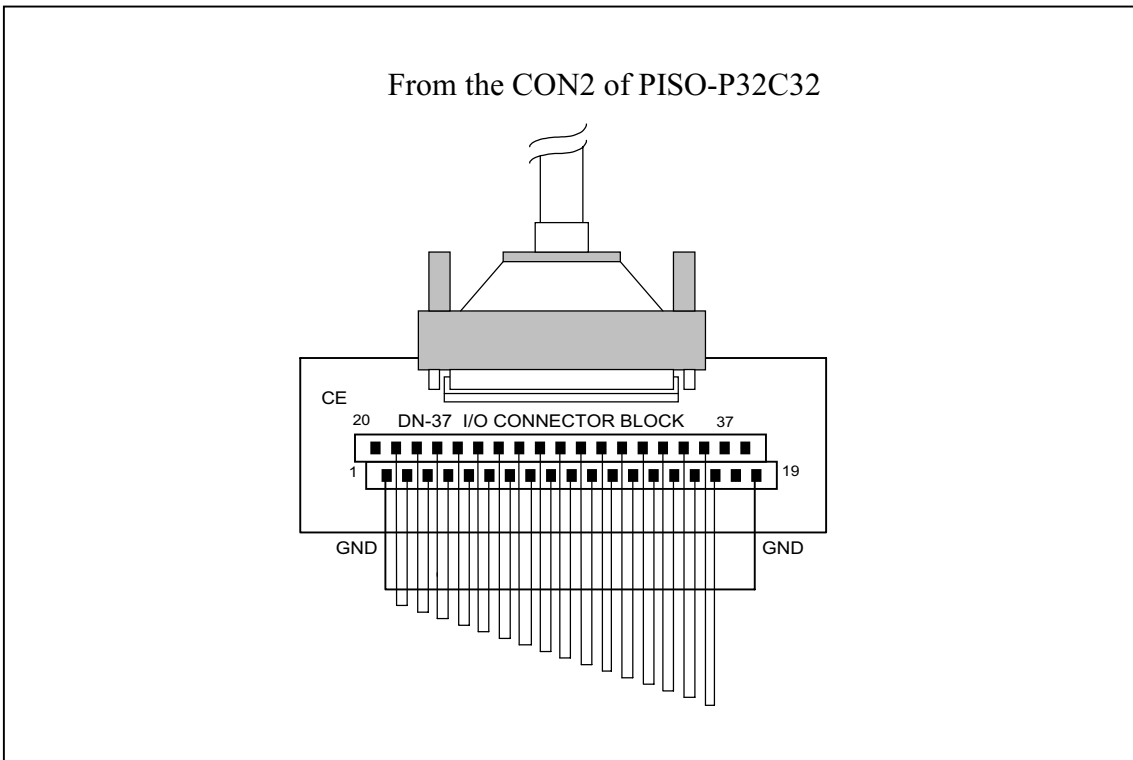


Figure 4-1-5. The D I/O of CON2 of PISO-P32C32

- The D/I of CON2 of PISO-P32C32 for internal power.

4.2 The example of PISO-P64

- The circuit diagram of D/I for PISO-P64 is given as follows:

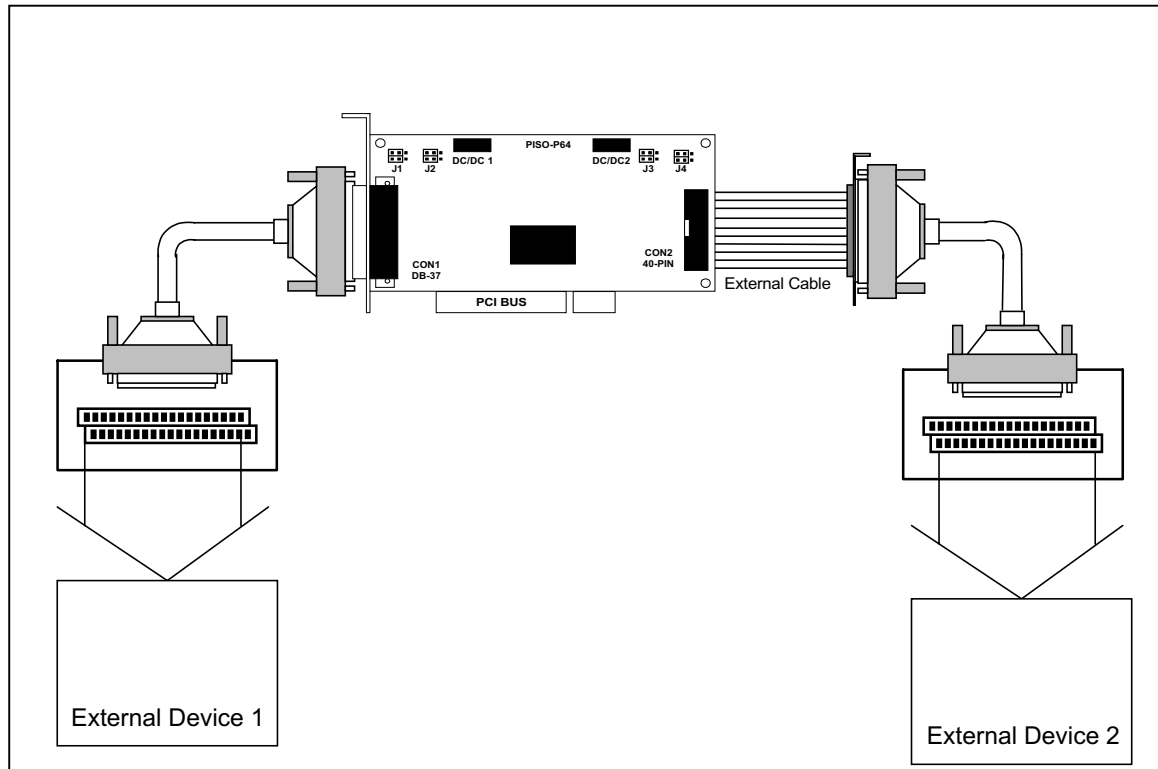
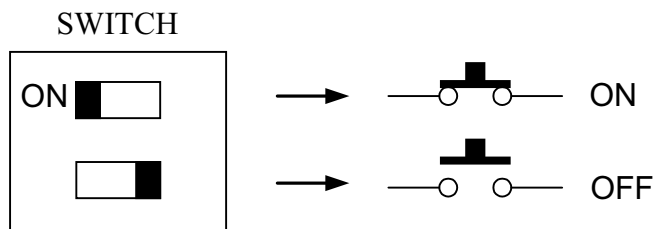


Figure 4-2-1. The example of digital inputs for PISO-P64

- The circuit diagram of external device 1: Refer to Figure 4-2-2
- The circuit diagram of external device 2: Refer to Figure 4-2-3



The circuit diagram of external device 1 is given as follows:

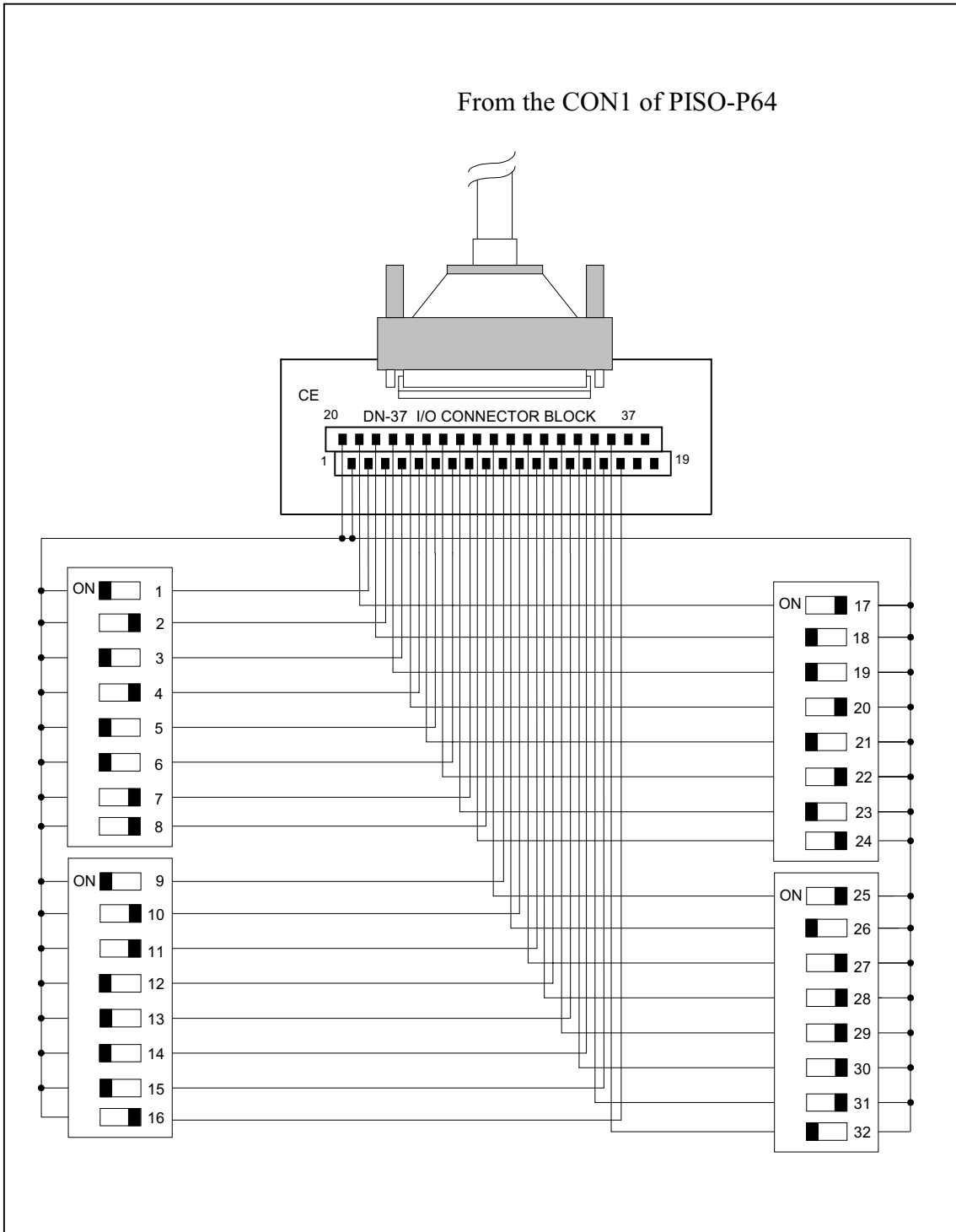


Figure 4-2-2. The circuit diagram of external device 2 for the digital inputs of PISO-P64

- The D/I of CON1 of PISO-P64 select internal power.

The circuit diagram of external device 2 is given as follows:

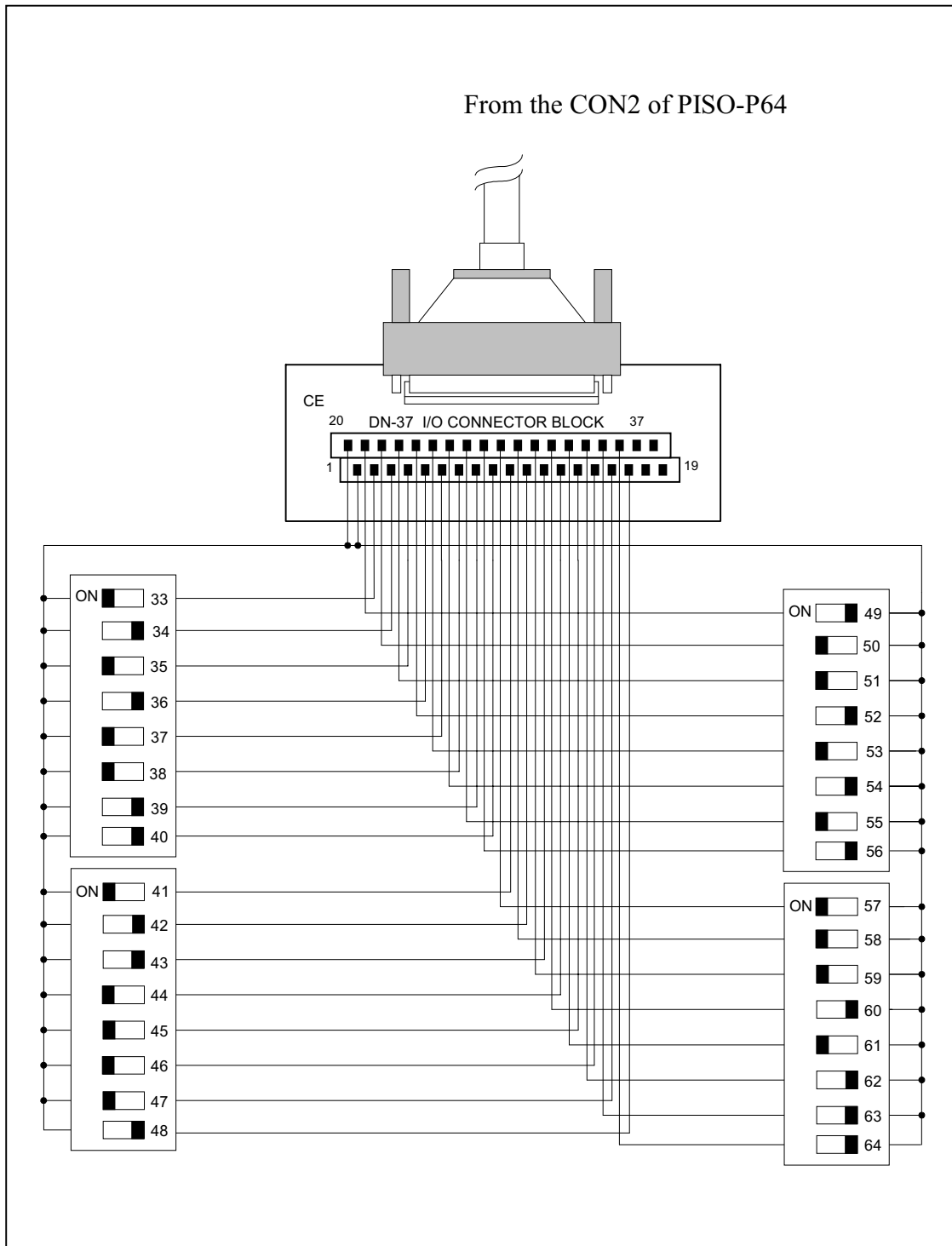


Figure 4-2-3. The circuit diagram of external device 2 for the digital inputs of PISO-P64

- The D/I of CON2 of PISO-P64 select internal power.

4.3 The example of PISO-C64

- The circuit diagram of D/O for PISO-P32C32 is given as follows:

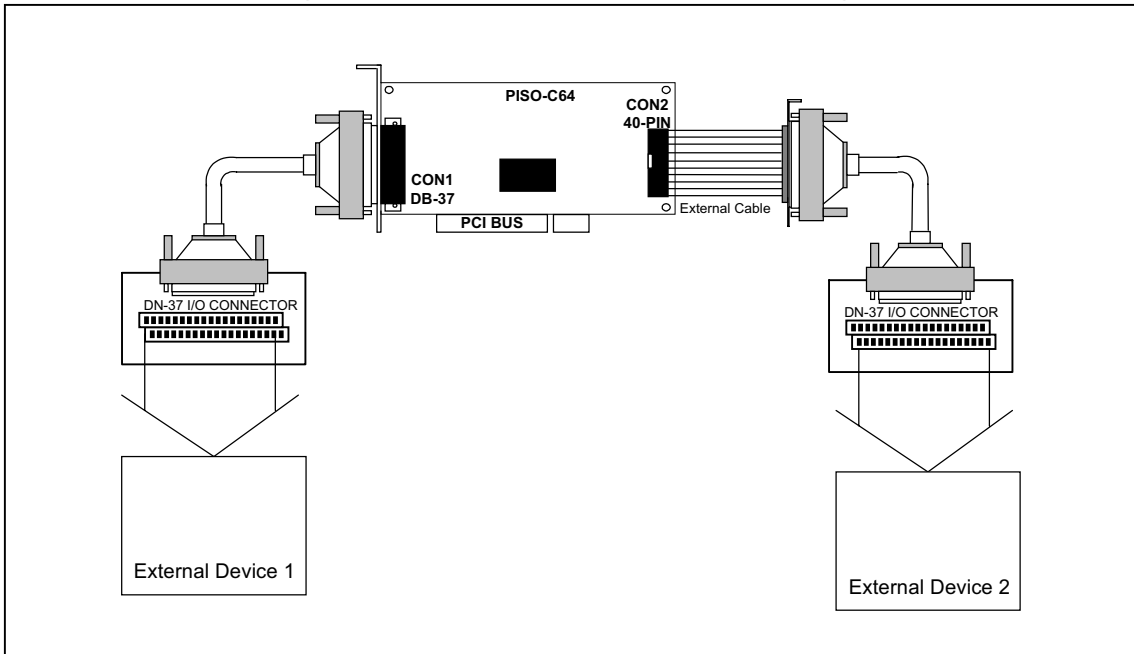


Figure 4-3-1. The example of digital outputs for PISO-C64

- The circuit diagram of external device 1: Refer to Figure 4-3-2
- The circuit diagram of external device 2: Refer to Figure 4-3-3

The circuit diagram of external device 1 is given as follows:

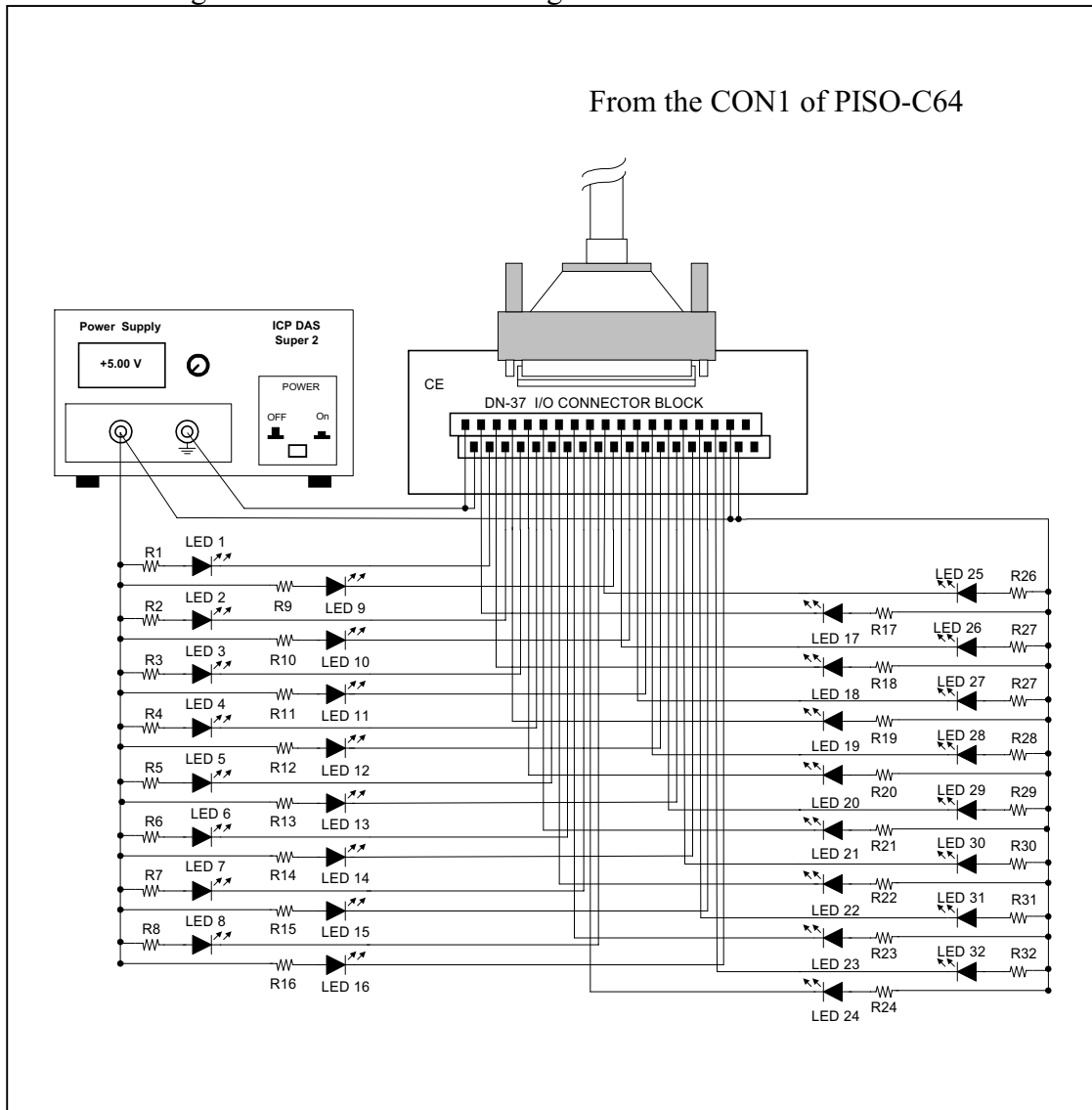


Figure 4-3-2. The circuit diagram of external device 1 for the digital outputs of PISO-C64

- The resistor of R1~R32 are 330 ohm.
- The LED 1~LED 32 are light emitting diode.
- The pin-1/20 are the GND signal of DO_0~DO_15 / DO_16~DO_31.
- The pin-18/37 are the voltage(+) signal of DO_0~DO_15 / DO_16~DO_31 (input DC +5V~+24V).

The circuit diagram of external device 2 is given as follows:

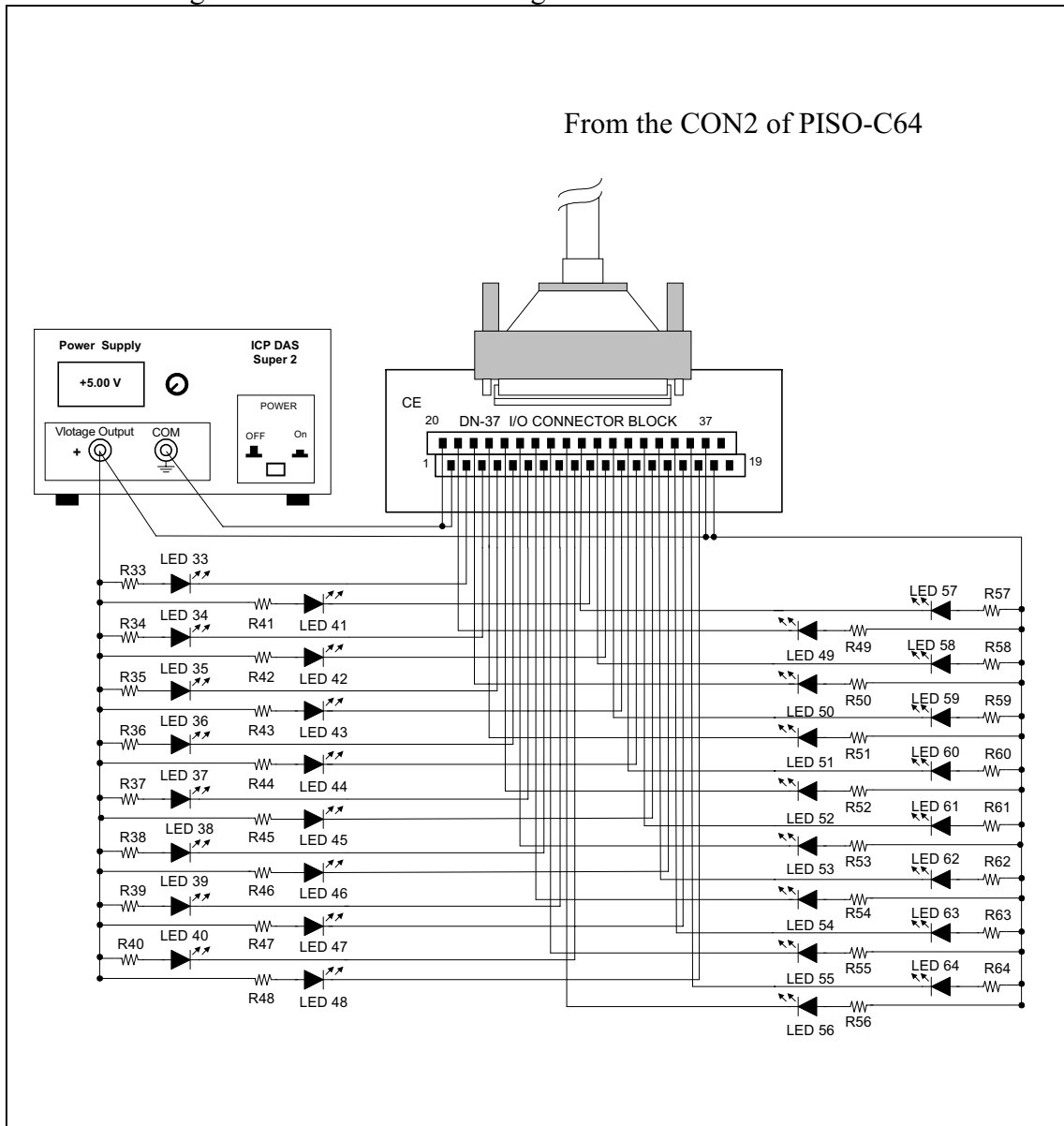


Figure 4-3-3. The circuit diagram of external device 2 for the digital outputs of PISO-C64

- The resistor of R33~R64 are 330 ohm.
- The LED 33~LED 64 are light emitting diode.
- The pin-1/20 are the GND signal of DO_32~DO_47 / DO_48~DO_63.
- The pin-18/37 are the voltage(+) signal of DO_32~DO_47 / DO_32~DO_63 (input DC +5V~+24V).

5. Demo Program

There are many demo program given in the company floppy disk or CD-ROM. After the software installation, the driver will be installed into disk as following:

5.1 Program file for PISO-P32C32

- ...\\P32C32\\TC>*. * → for Turbo C 2.xx or above
- ...\\P32C32\\BC>*. * → for Borland C++ 3.X above
- ...\\P32C32\\MSC>*. * → for Microsoft C 5.X above

- ...\\P32C32\\TC\\LIB>*. * → for library source code
- ...\\P32C32\\TC\\DEMO>*. * → demo program source code
- ...\\P32C32\\TC\\DIAG>*. * → pio_piso auto detect program

- ...\\P32C32\\TC\\LIB\\PIO.H → library header file
- ...\\P32C32\\TC\\LIB\\PIO.C → library source file
- ...\\P32C32\\TC\\LIB\\TCLIB.BAT → batch compiler file
- ...\\P32C32\\TC\\LIB\\TCPIO_L.LIB → I/O port large mode
- ...\\P32C32\\TC\\LIB\\TCPIO_H.LIB → I/O port huge mode
- ...\\P32C32\\TC\\LIB\\IOPORT_L.LIB → I/O port large mode
- ...\\P32C32\\TC\\LIB\\IOPORT_H.LIB → I/O port huge mode

- ...\\P32C32\\TC\\DEMO\\PIO.H → library header file
-\\P32C32\\TC\\DEMO\\DEMO1.C → demo1 source file
- ...\\P32C32\\TC\\DEMO\\DEMO2.C → demo2 source file
- ...\\P32C32\\TC\\DEMO\\DEMO3.C → demo3 source file
- ...\\P32C32\\TC\\DEMO\\DEMO1.PRJ → TC project1 file
- ...\\P32C32\\TC\\DEMO\\DEMO2.PRJ → TC project2 file
- ...\\P32C32\\TC\\DEMO\\DEMO3.PRJ → TC project3 file
- ...\\P32C32\\TC\\DEMO\\DEMO1.EXE → demo1 execution file
- ...\\P32C32\\TC\\DEMO\\DEMO2.EXE → demo2 execution file
- ...\\P32C32\\TC\\DEMO\\DEMO3.EXE → demo3 execution file

- ...\\P32C32\\TC\\DIAG\\PIO.H → library header file
- ...\\P32C32\\TC\\DIAG\\PIO_PSIO.C → I/O source code

- ...\\P32C32\\TC\\DIAG\\PIO_PISO.PRJ → TC project file
- ...\\P32C32\\TC\\DIAG\\PIO_PISO.EXE → I/O execution file

- ...\\P32C32\\BC\\LIB>*. * → for library source code
- ...\\P32C32\\BC\\DEMO>*. * → demo program source code
- ...\\P32C32\\BC\\DIAG>*. * → pio_piso auto detect program

- ...\\P32C32\\BC\\LIB\\PIO.H → library header file
- ...\\P32C32\\BC\\LIB\\PIO.C → library source file
- ...\\P32C32\\BC\\LIB\\BCLIB.BAT → batch compiler file
- ...\\P32C32\\BC\\LIB\\BCPIO_L.LIB → I/O port large mode
- ...\\P32C32\\BC\\LIB\\BCPIO_H.LIB → I/O port huge mode
- ...\\P32C32\\BC\\LIB\\IOPORT_L.LIB → I/O port large mode
- ...\\P32C32\\BC\\LIB\\IOPORT_H.LIB → I/O port huge mode

- ...\\P32C32\\BC\\DEMO\\PIO.H → library header file
- ...\\P32C32\\BC\\DEMO\\DEMO1.C → demo1 source file
- ...\\P32C32\\BC\\DEMO\\DEMO2.C → demo2 source file
- ...\\P32C32\\BC\\DEMO\\DEMO3.C → demo3 source file
- ...\\P32C32\\BC\\DEMO\\DEMO1.PRJ → BC project1 file
- ...\\P32C32\\BC\\DEMO\\DEMO2.PRJ → BC project2 file
- ...\\P32C32\\BC\\DEMO\\DEMO3.PRJ → BC project3 file
- ...\\P32C32\\BC\\DEMO\\DEMO1.EXE → demo1 execution file
- ...\\P32C32\\BC\\DEMO\\DEMO2.EXE → demo2.execution file
- ...\\P32C32\\BC\\DEMO\\DEMO3.EXE → demo3 execution file

- ...\\P32C32\\BC\\DIAG\\PIO.H → library header file
- ...\\P32C32\\BC\\DIAG\\PIO_PSIO.C → I/O source code
- ...\\P32C32\\BC\\DIAG\\PIO_PISO.PRJ → TC project file
- ...\\P32C32\\BC\\DIAG\\PIO_PISO.EXE → I/O execution file

- ...\\P32C32\\MSC\\LIB>*. * → for library source code
- ...\\P32C32\\MSC\\DEMO>*. * → demo program source code
- ...\\P32C32\\MSC\\DIAG>*. * → pio_piso auto detect program

- ...\\P32C32\\MSC\\LIB\\PIO.H → library header file
- ...\\P32C32\\MSC\\LIB\\PIO.C → library source file
- ...\\P32C32\\MSC\\LIB\\MSCLIB.BAT → batch compiler file

- ...P32C32\MSC\LIB\MSCPIO_L.LIB → I/O port large mode
- ...P32C32\MSC\LIB\MSCPIO_H.LIB → I/O port huge mode
- ...P32C32\MSC\LIB\IOPORT_L.LIB → I/O port large mode
- ...P32C32\MSC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...P32C32\MSC\DEMO\PIO.H → library header file
- ...P32C32\MSC\DEMO\DEMO1.C → demo1 source file
- ...P32C32\MSC\DEMO\DEMO2.C → demo2 source file
- ...P32C32\MSC\DEMO\DEMO3.C → demo3 source file
- ...P32C32\MSC\DEMO\MAKE1.BAT → demo1 batch file
- ...P32C32\MSC\DEMO\MAKE2.BAT → demo2 batch file
- ...P32C32\MSC\DEMO\MAKE3.BAT → demo3 batch file
- ...P32C32\MSC\DEMO\DEMO1.EXE → demo1 execution file
- ...P32C32\MSC\DEMO\DEMO2.EXE → demo2.execution file
- ...P32C32\MSC\DEMO\DEMO3.EXE → demo3 execution file

- ...P32C32\MSC\DIAG\PIO.H → library header file
- ...P32C32\MSC\DIAG\PIO_PSIO.C → I/O source code
- ...P32C32\MSC\DIAG\PIO.BAT → batch file
- ...P32C32\MSC\DIAG\PIO_PISO.EXE → I/O execution file

- :
- :
- :

5.2 Program file for PISO-P64

- ...\\P64\\TC>*. * → for Turbo C 2.xx or above
- ...\\P64\\BC>*. * → for Borland C++ 3.X above
- ...\\P64\\MSC>*. * → for Microsoft C 5.X above

- ...\\P64\\TC\\LIB>*. * → for library source code
- ...\\P64\\TC\\DEMO>*. * → demo program source code
- ...\\P64\\TC\\DIAG>*. * → pio_piso auto detect program

- ...\\P64\\TC\\LIB\\PIO.H → library header file
- ...\\P64\\TC\\LIB\\PIO.C → library source file
- ...\\P64\\TC\\LIB\\TCLIB.BAT → batch compiler file
- ...\\P64\\TC\\LIB\\TCPIO_L.LIB → I/O port large mode
- ...\\P64\\TC\\LIB\\TCPIO_H.LIB → I/O port huge mode
- ...\\P64\\TC\\LIB\\IOPORT_L.LIB → I/O port large mode
- ...\\P64\\TC\\LIB\\IOPORT_H.LIB → I/O port huge mode

- ...\\P64\\TC\\DEMO\\PIO.H → library header file
- ...\\P64\\TC\\DEMO\\DEMO1.C → demo1 source file
- ...\\P64\\TC\\DEMO\\DEMO1.PRJ → TC project1 file
- ...\\P64\\TC\\DEMO\\DEMO1.EXE → demo1 execution file

- ...\\P64\\TC\\DIAG\\PIO.H → library header file
- ...\\P64\\TC\\DIAG\\PIO_PSIO.C → I/O source code
- ...\\P64\\TC\\DIAG\\PIO_PISO.PRJ → TC project file
- ...\\P64\\TC\\DIAG\\PIO_PISO.EXE → I/O execution file

- ...\\P64\\BC\\LIB>*. * → for library source code
- ...\\P64\\BC\\DEMO>*. * → demo program source code
- ...\\P64\\BC\\DIAG>*. * → pio_piso auto detect program

- ...\\P64\\BC\\LIB\\PIO.H → library header file
- ...\\P64\\BC\\LIB\\PIO.C → library source file
- ...\\P64\\BC\\LIB\\BCLIB.BAT → batch compiler file
- ...\\P64\\BC\\LIB\\BCPIO_L.LIB → I/O port large mode
- ...\\P64\\BC\\LIB\\BCPIO_H.LIB → I/O port huge mode

-
- ...P64\BC\LIB\IOPORT_L.LIB → I/O port large mode
 - ...P64\BC\LIB\IOPORT_H.LIB → I/O port huge mode

 - ...P64\BC\DEMO\PIO.H → library header file
 - ...P64\BC\DEMO\DEMO1.C → demo1 source file
 - ...P64\BC\DEMO\DEMO1.PRJ → BC project1 file
 - ...P64\BC\DEMO\DEMO1.EXE → demo1 execution file

 - ...P64\BC\DIAG\PIO.H → library header file
 - ...P64\BC\DIAG\PIO_PSIO.C → I/O source code
 - ...P64\BC\DIAG\PIO_PISO.PRJ → BC project file
 - ...P64\BC\DIAG\PIO_PISO.EXE → I/O execution file

 - ...P64\MSC\LIB>*. * → for library source code
 - ...P64\MSC\DEMO>*. * → demo program source code
 - ...P64\MSC\DIAG>*. * → pio_piso auto detect program

 - ...P64\MSC\LIB\PIO.H → library header file
 - ...P64\MSC\LIB\PIO.C → library source file
 - ...P64\MSC\LIB\MSCLIB.BAT → batch compiler file
 - ...P64\MSC\LIB\MSCPIO_L.LIB → I/O port large mode
 - ...P64\MSC\LIB\MSCPIO_H.LIB → I/O port huge mode
 - ...P64\MSC\LIB\IOPORT_L.LIB → I/O port large mode
 - ...P64\MSC\LIB\IOPORT_H.LIB → I/O port huge mode

 - ...P64\MSC\DEMO\PIO.H → library header file
 - ...P64\MSC\DEMO\DEMO1.C → demo1 source file
 - ...P64\MSC\DEMO\MAKE1.BAT → demo1 batch file
 - ...P64\MSC\DEMO\DEMO1.EXE → demo1 execution file
 -
 - ...P64\MSC\DIAG\PIO.H → library header file
 - ...P64\MSC\DIAG\PIO_PSIO.C → I/O source code
 - ...P64\MSC\DIAG\PIO.BAT → batch file
 - ...P64\MSC\DIAG\PIO_PISO.EXE → I/O execution file

5.3 Program file list for PISO-C64

- ...\\C64\\TC>*. * → for Turbo C 2.xx or above
- ...\\C64\\BC>*. * → for Borland C++ 3.X above
- ...\\C64\\MSC>*. * → for Microsoft C 5.X above

- ...\\C64\\TC\\LIB>*. * → for library source code
- ...\\C64\\TC\\DEMO>*. * → demo program source code
- ...\\C64\\TC\\DIAG>*. * → pio_piso auto detect program

- ...\\C64\\TC\\LIB\\PIO.H → library header file
- ...\\C64\\TC\\LIB\\PIO.C → library source file
- ...\\C64\\TC\\LIB\\TCLIB.BAT → batch compiler file
- ...\\C64\\TC\\LIB\\TCPIO_L.LIB → I/O port large mode
- ...\\C64\\TC\\LIB\\TCPIO_H.LIB → I/O port huge mode
- ...\\C64\\TC\\LIB\\IOPORT_L.LIB → I/O port large mode
- ...\\C64\\TC\\LIB\\IOPORT_H.LIB → I/O port huge mode

- ...\\C64\\TC\\DEMO\\PIO.H → library header file
- ...\\C64\\TC\\DEMO\\DEMO1.C → demo1 source file
- ...\\C64\\TC\\DEMO\\DEMO1.PRJ → TC project1 file
- ...\\C64\\TC\\DEMO\\DEMO1.EXE → demo1 execution file

- ...\\C64\\TC\\DIAG\\PIO.H → library header file
- ...\\C64\\TC\\DIAG\\PIO_PSIO.C → I/O source code
- ...\\C64\\TC\\DIAG\\PIO_PISO.PRJ → I/O project file
- ...\\C64\\TC\\DIAG\\PIO_PISO.EXE → I/O execution file

- ...\\C64\\BC\\LIB>*. * → for library source code
- ...\\C64\\BC\\DEMO>*. * → demo program source code
- ...\\C64\\BC\\DIAG>*. * → pio_piso auto detect program

- ...\\C64\\BC\\LIB\\PIO.H → library header file
- ...\\C64\\BC\\LIB\\PIO.C → library source file
- ...\\C64\\BC\\LIB\\BCLIB.BAT → batch compiler file
- ...\\C64\\BC\\LIB\\BCPIO_L.LIB → I/O port large mode
- ...\\C64\\BC\\LIB\\BCPIO_H.LIB → I/O port huge mode

-
- ...C64\BC\LIB\IOPORT_L.LIB → I/O port large mode
 - ...C64\BC\LIB\IOPORT_H.LIB → I/O port huge mode

 - ...C64\BC\DEMO\PIO.H → library header file
 - ...C64\BC\DEMO\DEMO1.C → demo1 source file
 - ...C64\BC\DEMO\DEMO1.PRJ → BC project1 file
 - ...C64\BC\DEMO\DEMO1.EXE → demo1 execution file
 -
 - ...C64\BC\DIAG\PIO.H → library header file
 - ...C64\BC\DIAG\PIO_PSIO.C → I/O source code
 - ...C64\BC\DIAG\PIO_PISO.PRJ → BC project file
 - ...C64\BC\DIAG\PIO_PISO.EXE → I/O execution file

 - ...C64\MSC\LIB>*. * → for library source code
 - ...C64\MSC\DEMO>*. * → demo program source code
 - ...C64\MSC\DIAG>*. * → pio_piso auto detect program

 - ...C64\MSC\LIB\PIO.H → library header file
 - ...C64\MSC\LIB\PIO.C → library source file
 - ...C64\MSC\LIB\MSCLIB.BAT → batch compiler file
 - ...C64\MSC\LIB\MSCPIO_L.LIB → I/O port large mode
 - ...C64\MSC\LIB\MSCPIO_H.LIB → I/O port huge mode
 - ...C64\MSC\LIB\IOPORT_L.LIB → I/O port large mode
 - ...C64\MSC\LIB\IOPORT_H.LIB → I/O port huge mode

 - ...C64\MSC\DEMO\PIO.H → library header file
 - ...C64\MSC\DEMO\DEMO1.C → demo1 source file
 - ...C64\MSC\DEMO\MAKE1.BAT → demo1 batch file
 - ...C64\MSC\DEMO\DEMO1.EXE → demo1 execution file

 - ...C64\MSC\DIAG\PIO.H → library header file
 - ...C64\MSC\DIAG\PIO_PSIO.C → I/O source code
 - ...C64\MSC\DIAG\MAKE1.BAT → batch file
 - ...C64\MSC\DIAG\PIO_PISO.EXE → I/O execution file
 -
 -
 -

5.4 Diagnostic program

5.4.1 Diagnostic program for DOS

```
/* ----- */
/* Find all PIO_PISO series cards in this PC system */
/* step 1 : plug all PIO_PISO cards into PC */
/* step 2 : run PIO_PISO.EXE */
/* ----- */

#include "PIO.H"

WORD wBase,wIrq;
WORD wBase2,wIrq2;

int main()
{
int i,j,j1,j2,j3,j4,k,jj,dd,j11,j22,j33,j44;
WORD wBoards,wRetVal;
WORD wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
char c;
float ok,err;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0xff,0xff,0xff); /*for PIO-PISO*/
printf("\nThrer are %d PIO_PISO Cards in this PC",wBoards);
if (wBoards==0 ) exit(0);

printf("\n-----");
for(i=0; i<wBoards; i++)
{
PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,
&wSubDevice,&wSubAux,&wSlotBus,&wSlotDevice);

printf("\nCard_%d:wBase=%x,wIrq=%x,subID=[%x,%x,%x],
SlotID=[%x,%x]",i,wBase,wIrq,wSubVendor,wSubDevice,
wSubAux,wSlotBus,wSlotDevice);
printf(" --> ");
ShowPioPiso(wSubVendor,wSubDevice,wSubAux);
}

PIO_DriverClose();
}
```

NOTE: the PIO_PISO.EXE is valid for all PIO/PISO cards. The user can execute the PIO_PISO.EXE to get the following information:

- List all PIO/PISO cards installed in this PC
- List all resources allocated to every PIO/PISO cards
- List the wSlotBus & wSlotDevice for specified PIO/PISO card identification.

5.4.2 Diagnostic program for WINDOWS

There has an software utility “PIO_PISO.EXE” for Windows 95/98/NT for the detailed information about this file, please refer to the “Readme.txt” of development toolkit for Windows 95/98/NT .It is useful for all PIO/PISO series card.

- The setup step from the CD-ROM are given as follows:

Step 1: Toolkit (Softwares)/Manuals

Step 2: I AGREE

Step 3: PCI Bus DAQ Card

Step 4: PIO_PISO

Step 5: Install Toolkits for WINDOWS 98/98 or NT

After executing the utility, every detail information for all PIO/PISO cards that installed in the PC will be show as follows:



5.5 Demo program for PISO-P32C32

5.5.1 DEMO1 for PISO-P32C32

```
/* ----- */
/* Demo 1: Digital Output of PISO-P32C32 */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.1 */
/* Step 2: run demol.EXE */
/* ----- */

#include <dos.h>
#include "PIO.H"

int main()
{
char c;
BYTE i;
WORD wBoards,wRetVal;
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x20); /* for PISO-P32C32
*/
printf("\n(1) Threr are %d PISO-P32C32 Cards in this PC",wBoards);
if ( wBoards==0 )
{
putch(0x07); putch(0x07); putch(0x07);
printf("\n(1) There are no PISO-P32C32 card in this PC !!!");
exit(0);
}

printf("\n(2) The Configuration Space -> wBase");

for(i=0; i<wBoards; i++)
{

PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,&wSubDevice,&wSub
bAux,
&wSlotBus,&wSlotDevice);
printf("\nCard_%d:
wBase=%x,wIrq=%x,subID=[%x,%x,%x],SlotID=[%x,%x]"
,i,wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevi
ce);
}

/* step 1: enable all D/I/O port */
outportb(wBase,1); /* enable D/I/O */

/* step 2: Digital output from DO_0 to DO_31 */
while(1)
{
```

```

printf("\n\n ----- Digital output of PISO-P32C32 -----");
for (i=1;i<=0x80;i=i<<1)
{
    outportb(wBase+0xc0,i);      /* DO_07 to DO_00 */
    outportb(wBase+0xc4,i);      /* DO_15 to DO_08 */
    outportb(wBase+0xc8,i);      /* DO_23 to DO_16 */
    outportb(wBase+0xcc,i);      /* DO_31 to DO_24 */

    printf("\nD 31-0 Output Value = %02x,%02x,%02x,%02x",i,i,i,i);
    sleep(1);

    if(i==0x80) { i=0x01; break; }

    if (kbhit()!=0)
    {
        c=getch();
        if ((c=='q') || (c=='Q') || c==27 )
            return;
    }
    delay(1);
} /* end of while */
} /* end of for */

PIO_DriverClose();
}

```

5.5.2 DEMO2 for PISO-P32C32

```

/*-----*/
/* Demo 2: Digital input of PISO-P32C32 */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.1 */
/* Step 2: run demo2.EXE */
/* ----- */

#include <dos.h>
#include "PIO.H"

int main()

```

```
{
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
BYTE i,j1,j2,j3,j4;
char c;

WORD wBoards,wRetVal;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x20); /* for PISO-P32C32
*/
printf("\n(1) Threr are %d PISO-P32C32 Cards in this PC",wBoards);
if ( wBoards==0 )
{
    putchar(0x07); putchar(0x07); putchar(0x07);
    printf("\n(1) There are no PISO-P32C32 card in this PC !!!");
    exit(0);
}

printf("\n(2) The Configuration Space -> wBase");

for(i=0; i<wBoards; i++)
{

PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,&wSubDevice,&wSub
bAux,
                        &wSlotBus,&wSlotDevice);
    printf("\nCard_%d:
wBase=%x,wIrq=%x,subID=[%x,%x,%x],SlotID=[%x,%x]"
        ,i,wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevi
ce);
}

/* step 1: enable all D/I/O port */
outportb(wBase,1); /* enable D/I/O */

/* step 2: digital input from DI_0 to DI_31 */
while(1)
{
    for ( ; ; )
    {
        printf("\n\n ----- Digital input of PISO-P32C32 -----");
        j1=inportb(wBase+0xc0)^0xff; /* DI_07 to DI_00 */
        j2=inportb(wBase+0xc4)^0xff; /* DI_15 to DI_08 */
        j3=inportb(wBase+0xc8)^0xff; /* DI_23 to DI_16 */
        j4=inportb(wBase+0xcc)^0xff; /* DI_31 to DI_24 */

        printf("\nD 31-0 Input Value =
%02x,%02x,%02x,%02x",j4,j3,j2,j1);
        sleep(1);

        if(i==0x80) { i=0x01; break; }
    }
}
```

```
    if (kbhit()!=0)
    {
        c=getch();
        if ((c=='q') || (c=='Q') || c==27 )
            return;
    }
    delay(1);
} /* end of while */
} /* end of for */

PIO_DriverClose();
}
```

5.5.3 DEMO3 for PISO-P32C32

```
/* ----- */
/* Demo 3: Digital I/O test of PISO-P32C32          */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.1 */
/* Step 2: run demo3.EXE                            */
/* ----- */

#include <dos.h>
#include "PIO.H"

int main()
{
    WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
    BYTE i,j1,j2,j3,j4;
    char c;

    WORD wBoards,wRetVal;

    clrscr();
```

```
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x20); /* for PISO-P32C32
*/
printf("\n(1) Threr are %d PISO-P32C32 Cards in this PC",wBoards);
if ( wBoards==0 )
{
    putch(0x07); putch(0x07); putch(0x07);
    printf("\n(1) There are no PISO-P32C32 card in this PC !!!");
    exit(0);
}

printf("\n(2) The Configuration Space -> wBase");

for(i=0; i<wBoards; i++)
{

PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,&wSubDevice,&wSub
bAux,
                        &wSlotBus,&wSlotDevice);
    printf("\nCard %d:
wBase=%x,wIrq=%x,subID=[%x,%x,%x],SlotID=[%x,%x]"
        ,i,wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevi
ce);
}

/* step 1: enable all D/I/O port */
outportb(wBase,1); /* enable D/I/O */

/* step 2: DO_0 to DO_31 send to DI_0 to DI_31 */
while(1)
{
printf("\n\n ----- PISO-P32C32 test by itself -----");
for (i=1;i<=0x80;i=i<<1)
{

    outportb(wBase+0xc0,i); /* DO_07 to DO_00 */
    outportb(wBase+0xc4,i); /* DO_15 to DO_08 */
    outportb(wBase+0xc8,i); /* DO_23 to DO_16 */
    outportb(wBase+0xcc,i); /* DO_31 to DO_24 */
    delay(1); /* about to wait 1m sec */
    j1=inportb(wBase+0xc0)^0xff; /* DI_07 to DI_00 */
    j2=inportb(wBase+0xc4)^0xff; /* DI_15 to DI_08 */
    j3=inportb(wBase+0xc8)^0xff; /* DI_23 to DI_16 */
    j4=inportb(wBase+0xcc)^0xff; /* DI_31 to DI_24 */

printf("\nD 31-0 Output Value = %02x,%02x,%02x,%02x",i,i,i,i);
printf("\nD 31-0 Input Value = %02x,%02x,%02x,%02x\n",j4,j3,j2,j1);

if( i != j1 )
{
    printf("\nD I/O 7-0 error in here!\n");
    putch(0x07); putch(0x07); putch(0x07);
}
if( i != j2 )
{
    printf("\nD I/O 15-8 error in here!\n");
    putch(0x07); putch(0x07); putch(0x07);
}
if( i != j3 )
{
    printf("\nD I/O 24-16 error in here!\n");
    putch(0x07); putch(0x07); putch(0x07);
}
if( i != j4 )
{
    printf("\nD I/O 31-25 error in here!\n");
    putch(0x07); putch(0x07); putch(0x07);
}
}
}
}
```

```

    }

    if(i==j1 & i==j2 & i==j3 & i==j4)
    {
        printf("The Digital I/O test of PISO-P32C32 by itself OK!\n");
    }

    if(i==0x80) { i=0x01; break; }

    if (kbhit()!=0)
    {
        c=getch();
        if ((c=='q') || (c=='Q') || c==27 )
            return;
    }
    delay(1);
} /* end of while */
} /* end of for */

PIO_DriverClose();
}

```

5.6 Demo program for PISO-P64

5.6.1 DEMO1 for PISO-P64

```

/* ----- */
/* Demo 1: Digital Input of PISO-P64 */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.2 */
/* Step 2: run demol.EXE */
/* ----- */

#include <dos.h>
#include "PIO.H"

int main()
{
    char c;
    BYTE i,r1,r2,r3,r4,r5,r6,r7,r8;
    WORD wBoards,wRetVal;
    WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;

    clrscr();
    wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x10); /* for PISO-P64 */
    printf("\n(1) Threr are %d PISO-P64 Cards in this PC",wBoards);
    if ( wBoards==0 )
    {
        putch(0x07); putch(0x07); putch(0x07);
        printf("\n(1) There are no PISO-P64 card in this PC !!!");
    }
}

```

```
    exit(0);
}

printf("\n(2) The Configuration Space ->   wBase");

for(i=0; i<wBoards; i++)
{

PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,&wSubDevice,&wSubAux,
                          &wSlotBus,&wSlotDevice);
    printf("\nCard_%d:
wBase=%x,wIrq=%x,subID=[%x,%x,%x],SlotID=[%x,%x]"
          ,i,wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice);
}

/* step 1: enable all D/I/O port */
outportb(wBase,1);          /* enable D/I/O */

/* step 2: Digital input from DI_0 to DI_63 */
while(1)
{
for ( ; ; )
{
    printf("\n----- Digital input of PISO-P64 -----");
    r1 =inportb(wBase+0xc0);    /* DI_07 to DI_0  */
    r2 =inportb(wBase+0xc4);    /* DI_15 to DI_08 */
    r3 =inportb(wBase+0xc8);    /* DI_23 to DI_16 */
    r4 =inportb(wBase+0xcc);    /* DI_31 to DI_24 */

    r5 =inportb(wBase+0xd0);    /* DI_39 to DI_32 */
    r6 =inportb(wBase+0xd4);    /* DI_47 to DI_40 */
    r7 =inportb(wBase+0xd8);    /* DI_55 to DI_48 */
    r8 =inportb(wBase+0xdc);    /* DI_63 to DI_56 */

    printf("\nThe CON1 of PISO-P64 ...");
    printf("\nD31- 0 Input Value =
%02x,%02x,%02x,%02x\n",r4,r3,r2,r1);

    printf("\nThe CON2 of PISO-P64 ...");
    printf("\nD63-32 Input Value =
%02x,%02x,%02x,%02x\n",r8,r7,r6,r5);
    sleep(1);

    if (kbhit()!=0)
    {
        c=getch();
        if ((c=='q') || (c=='Q') || c==27 )
            return;
    }
    delay(1);
} /* end of while */
} /* end of for */

PIO_DriverClose();
}
```

5.7 Demo program for PISO-C64

5.7.1 DEMO1 for PISO-C64

```

/* ----- */
/* Demo 1: Digital Output of PISO-C64 */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.3 */
/* Step 2: run demol.EXE */
/* ----- */

#include <dos.h>
#include "PIO.H"

int main()
{
char c;
BYTE i;
WORD wBoards,wRetVal;
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x00); /* for PISO-C64 */
printf("\n(1) Threr are %d PISO-C64 Cards in this PC",wBoards);
if ( wBoards==0 )
{
putch(0x07); putch(0x07); putch(0x07);
printf("\n(1) There are no PISO-C64 card in this PC !!!");
exit(0);
}

printf("\n(2) The Configuration Space -> wBase");

for(i=0; i<wBoards; i++)
{

PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,&wSubDevice,&wSu
bAux,

```

```
        &wSlotBus,&wSlotDevice);
    printf("\nCard_%d:
wBase=%x,wIrq=%x,subID=[%x,%x,%x],SlotID=[%x,%x]"
        ,i,wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevi
ce);
    }

/* step 1: enable all D/I/O port */
outportb(wBase,1); /* enable D/I/O */

/* step 2: Digital output from DO_0 to DO_63 */
while(1)
{
printf("\n\n ----- Digital output of PISO-C64 -----");
for (i=1;i<=0x80;i=i<<1)
    {
        outportb(wBase+0xc0,i); /* DO_07 to DO_00 */
        outportb(wBase+0xc4,i); /* DO_15 to DO_08 */
        outportb(wBase+0xc8,i); /* DO_23 to DO_16 */
        outportb(wBase+0xcc,i); /* DO_31 to DO_24 */
        outportb(wBase+0xd0,i); /* DO_39 to DO_32 */
        outportb(wBase+0xd4,i); /* DO_47 to DO_40 */
        outportb(wBase+0xd8,i); /* DO_55 to DO_48 */
        outportb(wBase+0xdc,i); /* DO_63 to DO_56 */

        printf("\nThe CON1 of PISO-C64 ...");
        printf("\nD31- 0 Output Value = %02x,%02x,%02x,%02x\n",i,i,i,i);

        printf("\nThe CON2 of PISO-C64 ...");
        printf("\nD63-32 Output Value = %02x,%02x,%02x,%02x\n",i,i,i,i);
        sleep(1);
    }

if(i==0x80) { i=0x01; break; }

    if (kbhit()!=0)
    {
        c=getch();
        if ((c=='q') || (c=='Q') || c==27 )
            return;
    }
    delay(1);
} /* end of while */
} /* end of for */

PIO_DriverClose();
}
```